HP Integrity Servers with Microsoft Windows Server 2003 for Itanium-based systems SmartSetup Scripting Toolkit Deployment Guide

HP Part Number: 5992-0954
Published: December 2007

© Copyright 2007 Hewlett-Packard Development Company, L.P

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

The information contained herein is subject to change without notice. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein

SmartSetup Scripting Toolkit uses Linux and related open source software under the terms of the GPL and includes the source of that software in the SmartSetup Scripting Toolkit package.

Microsoft and Windows are U.S. registered trademarks of Microsoft Corporation. Intel and Itanium are registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries. UNIX is a registered trademark of The Open Group.

Table of Contents

About This Document....13
Intended Audience....13
Typographic Conventions....13
HP Encourages Your Comments....13

1 Overview....15
Introduction....15
Toolkit components....16
Plan the deployment....17
Select a workstation....17
Select a boot mechanism....17
Position the repository....18
Go with the workflow....18
Setting up a model server....18
Creating a server profile....19
Setting up the toolkit environment....19
Launching the server replication....19
Customizing the toolkit environment....20
Troubleshooting....20
Log file format....20

2 Creoting o server profile....23
Prepare the workstation....23
Display headless server console output at the workstation....23
Unpack the toolkit....25
Setting permissions for repository access....29
CLM mode check....30
Upload the operating system installation files....30
Set up an unattended operating system installation....32
Accessing the server after a Windows installation....33
Installing tools from the Integrity Support Pack....34
Copying software to the HPSP partition....34
Upload the device driver files....34
Set up driver installation....39

3 Setting up the toolkit environment....41
Edit sstk.conf....41
Edit bootia64.conf....42
Edit elilo.conf....42
Review main_toolkit_launch.sh....45
Edit windows-setup.sh....46

4 Setting up the boot mechonism....49
Using a bootable CD/DVD....49
From a Windows workstation....49
Create a bootable disk image....49
Create the ISO using Roxio....50
Create the ISO using Nero....52

From a Linux workstation....54
Create a bootable disk image....54
Create a CD build directory....54
Create the ISO file....55
Booting from the CD/DVD....55
Using a USB flash device....55
Create a bootable USB flash device on a Windows workstation....55
Create a bootable USB flash device on a Linux workstation....56
Boot from the USB flash device....56
Using USB flash device as external media....57
Using network boot....57
Setting up a Windows network boot server....58
Setting up a TFTP server....58
Setting up a DHCP server....58
Setting up a Linux network boot Server....58
Setting up a DHCP server....58
Setting up a TFTP server....59
Populating the TFTP directory share....59
Setting up the Integrity server for PXE-enabled boot....60
Setting up a Cellular-based Integrity server for PXE-enabled boot....60
Booting the Integrity server from a PXE-enabled NIC....60
Interrupting the SSTK deployment process....60

5 Utilities Reference....63
cfggen....63
Synopsis....63
Description....63
Options....63
Examples....63
CLMChecking....65
Synopsis....65
Description....65
Options....65
Example....65
conrep....66
Synopsis....66
Description....66
Options....66
Examples....66
Return codes....67
cpqacuxe....68
Synopsis....68
Description....68
cpqacuxe capture file....68
Options....69
cpqacuxe input file....69
Automatic script file format....70
Custom script file format....72
Error reporting....78
efibootmgr....83
Synopsis....83
Description....83
Options....83
efivar....85

Synopsis....85
Description....85
Options....85
Example....85
gen-setupldr-options....86
Synopsis....86
Description....86
Options....86
getuuid....87
Synopsis....87
Description....87
Options....87
hpia64info....88
Synopsis....88
Description....88
Options....88
Output....88
hplpcfg....89
Synopsis....89
Description....89
Options....89
Format....89
Example....89
hpqlarep....90
Synopsis....90
Description....90
Options....90
Format....90
hpsetacpi....91
Synopsis....91
Description....91
Options....91
Example....91
hpshowdisk....92
Synopsis....92
Description....92
Options....92
Example....92
hwdisc....94
Synopsis....94
Description....94
Options....94
Example....94
hwquery....96
Synopsis....96
Description....96
Options....96
Examples....96
ifhw....97
Synopsis....97
Description....97
Options....97
Examples....97
parted....99
Synopsis....99

Description....99
Options....99
Examples....99
sstkencrypt....101
Synopsis....101
Description....101
Example....101

A Support Matrix....103
Supported systems....103
Supported LSI and SmartArray SCSI/RAID controllers....103
Supported Emulex and QLogic Fiber Channel controllers....104
Supported Intel, Broadcom, and Neterion NIC I/O cards....104
Supported storage....105

List of Figures

1-1 Clone a Server Setup....15
1-2 Main Components of SSTK Deployment....16
1-3 Linux Boot Files....17
2-1 Console Configuration Window....24
2-2 Console Output Devices Window....24
2-3 SSTK Archive....26
2-4 /os/windows/2003-enterprise/ Directory....31
4-1 EFI Directory....50
4-2 Disc Creator Classic Main Window....51
4-3 Roxio Project Area....51
4-4 Create Disc Image....52
4-5 New Compilation Window....53
4-6 Burn Tab....53
4-7 Copy Boot files....55
4-8 EFI shell Window....56
4-9 Boot Configuration Selection Window....57
4-10 Boot from Linux USB Device Window....57
5-1 hpshowdisk Utility....93

List of Tables

2-1 SSTK Repository Permissions....29
3-1 Global options....42
3-2 Image options....43
3-3 SSTK command line options....43
4-1 `mkisofs` command options....55
5-1 cfggen Command Values....63
5-2 CLMChecking Options....65
5-3 conrep Options....66
5-4 conrep Return Codes....67
5-5 cpqacuxe Options....69
5-6 Automatic script file options....70
5-7 Custom script file options....73
5-8 cpqacuxe Error Codes....79
5-9 efibootmgr Options....83
5-10 Commands....83
5-11 efibootmgr Command Line Input....84
5-12 gen-setupldr-options Options....86
5-13 gen-setupldr-options Command line input....86
5-14 hplpcfg Options....89
5-15 hpsetacpi Options....91
5-16 hpshowdisk Options....92
5-17 hwdisc Options....94
5-18 hwquery Options....96
5-19 ifhw Options....97
5-20 ifhw Operators....98
5-21 parted Options....99
A-1 SCSI Controllers....103
A-2 ISS Controllers....103
A-3 QLogic controllers....104
A-4 Emulex controllers....104
A-5 Intel cards....104
A-6 Broadcom cards....104
A-7 Neterion cards....104

List of Examples

5-1 cfggen list command....63
5-2 cfggen display command....63
5-3 Sample conrep.dat file....66
5-4 conrep.xml file....66
5-5 Sample hpia64info utility output....88

About This Document

This document describes how to use the SmartSetup Scripting Toolkit (SSTK) to develop custom scripts that simplify server deployments by automating various hardware configuration and software installation operations.

Intended Audience

This document is for system administrators who need to deploy a large number of HP Integrity® servers rapidly and efficiently.

Typographic Conventions

This document uses the following typographical conventions:

`Command`	A command name or qualified command phrase.
`Computer output`	Text displayed by the computer.
Ctrl+x	A key sequence. A sequence such as **Ctrl+x** indicates that you must hold down the key labeled **Ctrl** while you press another key or mouse button.
`File Name`	The name of a file name or path, for example, `/os/windows`.
Key	The name of a keyboard key. **Return** and **Enter** both refer to the same key.
Term	The defined use of an important word or phrase.
`User input`	Commands and other text that you type.
`Variable`	The name of a placeholder in a command, function, or other syntax display that you replace with an actual value.

HP Encourages Your Comments

HP encourages your comments concerning this document. We are committed to providing documentation that meets your needs. Send any errors found, suggestions for improvement, or compliments to:

feedback@fc.hp.com

Include the document title, manufacturing part number, and any comment, error found, or suggestion for improvement you have concerning this document.

1 Overview

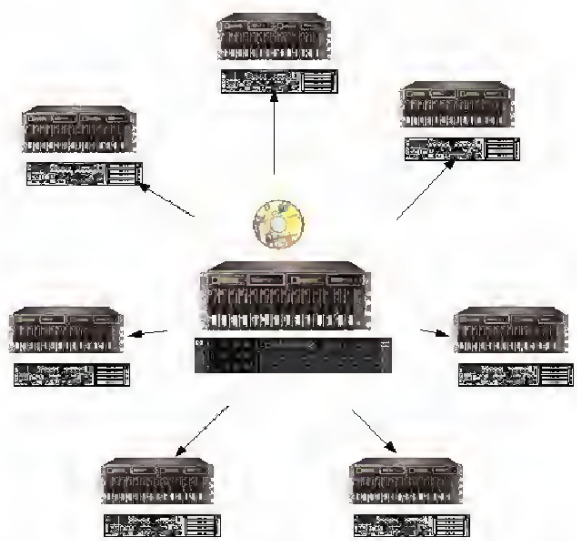
The SmartSetup Scripting Toolkit (SSTK) enables you to deploy a large number of HP Integrity® servers rapidly and efficiently. Using SSTK, you can develop custom scripts that simplify server deployments by automating various hardware configuration and software installation operations. SSTK can set specific EFI boot variables, create disk partitions, and tie into the standard unattended installation process to install the OS and selected applications.

Introduction

When a widget needs to be mass produced, the usual trick is to create a model of the widget and then clone it using an automated process. To produce a thousand CDs, for example, a CD manufacturer first creates a **master** CD and then copies it a thousand times using a replication process. In the movies, a mad scientist extracts the DNA from a model subject and, with some spectacular special effects, clones an army of replicants. Although you are unlikely to be in the business of replicating CDs or people, your approach to deploying a large number of servers is likely to be similar.

To deploy a hundred identical HP Integrity servers quickly and efficiently, you would set up a model server manually and then clone this setup (Figure 1-1) on the remaining ninety nine servers using an automated replication process. Without automation, the deployment of the hundredth server would be just as time-consuming, laborious, and error-prone as the first. But by using an automated replication process, you would significantly reduce the time, effort, and errors involved in such high-volume server deployments.

Figure 1-1 Clone a Server Setup



Automated server replication is the basis for the Smart Setup Scripting Toolkit. The SSTK enables you to develop a process of replicating the hardware configuration of a model server and attached storage array controllers. The SSTK also ties into standard, unattended installation processes for the Windows® OS and applications.

Server deployments using SSTK involve four main stages: First, you set up a model Integrity server. Next, you create a server profile based on the model server. Then, you create a scripted process to apply the server profile onto a target server. Finally, you launch this process to replicate the server profile on hundreds of servers.

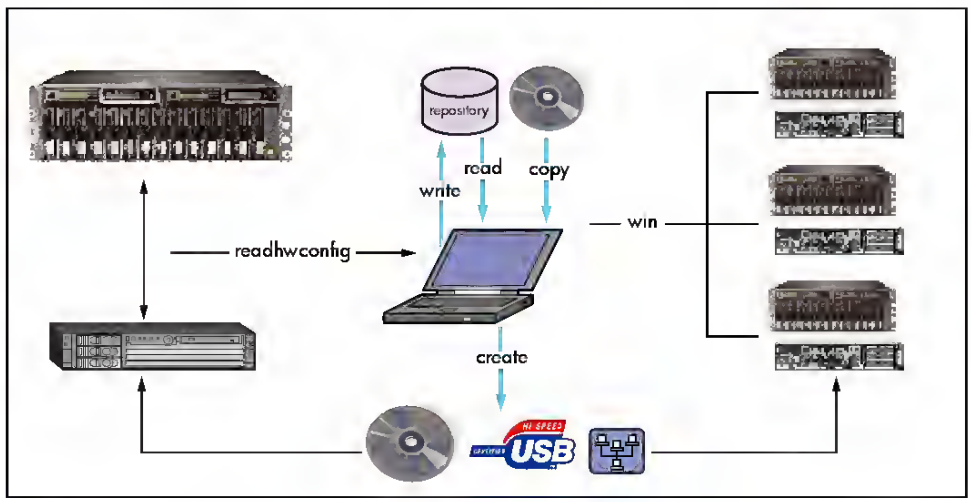
That said, the SSTK is not an automated server deployment solution for everyone. The SSTK is an extensible framework for developing hardware configuration and software installation scripts. The SSTK is designed for administrators who have some expertise in unattended installations of Windows Server 2003 and familiarity with Integrity servers. The SSTK provides sample scripts and utilities that run in a staging environment based on Debian Linux for IA-64. The kit uses Linux merely as a launch pad for unattended server configuration and Windows installation. You do not need to know Linux well to use the toolkit but you must be able to read and modify shell scripts.

This document guides you step-by-step through the typical tasks involved in deploying HP Integrity servers using the SSTK. It also provides some guidance on customizing the toolkit for your deployment scenario. You will need to extend the samples provided in the toolkit to take advantage of these capabilities. We recommend that you read this book completely before planning your deployment. And always remember to validate your scripts on a test target server before executing them on the remaining ninety eight.

Toolkit components

Figure 1-2 illustrates the main components of an SSTK-based deployment.

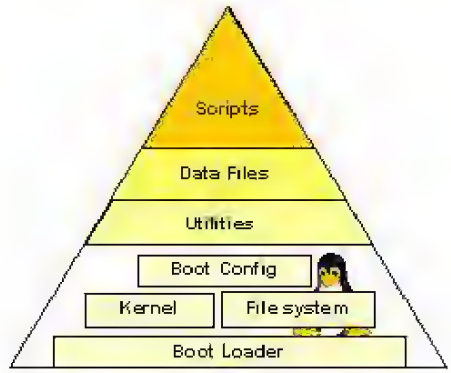
Figure 1-2 Main Components of SSTK Deployment



- The **model server** sets the standard for the configuration of all other servers. You can use an existing server as the model or set up a new one.
- The configuration of the **storage array controllers** installed in the model server sets the standard for all others. You can configure the controllers using the HP Option ROM Configuration for Arrays (ORCA) EFI application before booting the OS or run the Array Configuration Utility (ACU) after the OS is installed.
- The **workstation** — a PC equipped with a CD- or DVD-write drive running Windows or Linux — is the base of SSTK operations. You need the workstation to download SSTK, unpack the software package, edit the sample scripts, and create a bootable device. You can also use the workstation to host a **repository** of custom scripts, utilities, hardware configuration files, OS files, and application files.
- The **operating system CD** — specifically, the Microsoft® Windows Server™ 2003 Enterprise Edition for Itanium-based Systems Release To Manufacturing (RTM) CD — contains the OS setup files. You must have a volume license that allows you to copy this CD and install this OS on multiple servers in your deployment.
- The **SSTK software** package provides sample scripts that invoke utilities, which read and modify the server and storage array controller configuration. The scripts copy data files (configuration files), along with OS and application files you supply, on to the target servers.

The utilities run in a Linux environment. The kit provides Linux boot files (boot loader, kernel, root file system, and boot configuration) for you to copy onto the boot device (Figure 1-3).

Figure 1-3 Linux Boot Files



- The **SSTK boot device** — a CD or DVD, a network share, or a USB flash drive — holds files that allow servers to boot to the toolkit environment. You can also use the SSTK boot device to hold the repository of scripts, utilities, configuration files, OS files, and application files.
- The **target servers** are the machines you want to set up as replicas.

Plan the deployment

The following sections provide guidance on the various ways in which you can set up these three components. The server replication process depends on what you select here.

Select a warkstatian

The **workstation** is any recent PC, equipped with a CD writer or DVD writer (and in some cases a USB or PXE capability), running any version of Windows or any distribution of Linux. You will use the workstation to download the SSTK, unpack the SSTK software package, edit the sample scripts, and create a bootable device. You can also use the workstation to host the repository of custom scripts, utilities, hardware configuration files, OS files, and application files.

You must select to use either a Windows workstation or a Linux workstation. If you plan to do extensive customization of either the scripts or the Linux boot files in the toolkit environment, consider using a Linux workstation. If not, a Windows workstation may fit more easily into your usual workspace.

Select a baat mechanism

The SSTK supports three **boot mechanisms** — CD/DVD, USB flash or network boot via a PXE-enabled NIC. The boot mechanism holds the files that boot the server to the toolkit environment and run the main script. The main script gets configuration files, OS files, and application files from the repository, configures the hardware, and installs the software. You can also use the DVD, a USB or network share to host the repository, which would gather all runtime components in one place.

Several factors affect your choice of the boot mechanism:

- **Capacity:** The choice between the CD and DVD is function of the capacity of the disk. You may need a DVD if you plan to host the repository (including the OS files along with the configuration files, scripts, and utilities) on the same device as the boot files (toolkit environment). In particular, the Japanese edition of the OS and the English edition with

slipstreamed SP1 both exceed the capacity of a CD that already contains the toolkit environment.

- **Flexibility:** The choice between the CD and the USB flash drive boils down to the flexibility that you need from the toolkit environment and custom scripts. You can quickly edit the Linux boot loader configuration file, a script, or configuration file on a USB flash drive but burning a new CD is more complicated.
- **Complexity:** The choice between a network boot and a boot device depends on the scope and complexity of your deployment. If you plan to set up over a hundred servers at a time, you should definitely invest in a PXE-enabled installation environment in which the Integrity servers boot over the network. On the other hand, if you plan to deploy your servers in more measured increments, the DVD may be an optimal solution. You could burn a dozen DVDs at a time, insert each in an Integrity server, reboot, and return to a dozen managed servers.

Position the repository

Every server that boots to the SSTK environment must have access to the repository of configuration files, scripts, utilities, OS files, driver files, and application files. The Linux boot loader configuration file specifies the location of the repository.

If you host the repository on the workstation, it must be accessible to the Integrity servers being deployed. If, for example, your workstation is on a public network and the servers are on a private isolated network, the SSTK will not work. If you host the repository on a boot device, be sure the device has enough capacity to hold both the SSTK environment and the OS.

IMPORTANT: A repository solution is not limited to the previous example. You can develop a hybrid solution if that works for you. For example, you can boot the model server from a USB device, host the repository on the workstation to read the configuration of the model server, customize the scripts and files, create a bootable DVD for testing on a few servers, and then set up a network-based installation for a full-scale deployment.

Go with the workflow

Server deployments using SSTK involve four main stages. The following subsections provide a high level overview of the deployment workflow.

Setting up a model server

If you do not have a preconfigured Integrity server similar to those you want to deploy, select one of the servers to be deployed and configure it as the model server. On cell-based servers you must configure the hard partitions on the model server using Partition Manager. For detailed instructions, see the nPartition Guide (on the Smart Setup CD or at http://docs.hp.com/en/windows.html). You can configure the server using the EFI-based Setup Utility on the Smart Setup CD. You can configure the storage array controller using the EFI-based Option ROM Configuration for Arrays (ORCA) before booting the OS or run the Array Configuration Utility (ACU) after the OS is installed. For detailed instructions, see the Smart Setup Guide (on the Smart Setup CD or at http://docs.hp.com/en/windows.html).

The Smart Setup CD is included in the HP Integrity Essentials Foundation Pack Kit. You can also download the ISO from http://www.hp.com/support/itaniumservers/. You will also need the Microsoft Windows Server 2003 Enterprise Edition CD.

NOTE: SSTK does not support Microsoft Windows Server 2003 Datacenter Edition or HP Reinstallation media.

Creating a server profile

A server profile is like the DNA of all servers in the deployment. It includes the server EFI environment variables, the I/O configuration, the OS and drivers, and installed applications. The server profile is stored in the repository. Creating it involves the following tasks:

1. **Prepare the workstation.** On a Windows workstation, ensure that you have a text editor (`crimson`, `textpad`, or `vim`; See "Prepare the workstation" (page 23)) that can save files in the UNIX® file format or a DOS to UNIX file converter. Install applications that can create a disk image (`WinImage`) and burn a bootable CD or DVD (`Nero` or `Roxio`).
2. **Unpack the toolkit.** Download the SSTK software package and expand it on the workstation. The directory structure of the software package is important. It forms the framework of the repository and is designed to help you set up the server profile. The scripts expect this structure. Although you can change the scripts, retaining this structure will save you some customization effort.
3. **Set up the unattended OS installation.** The SSTK ties into the standard OS unattended installation process. Copy the OS files from the WS2003 EE CD to the workstation into the repository. SSTK provides a sample `\os\unattended.txt` answer file, which supplies Setup with answers to questions that you are asked during an interactive installation. Edit the file to enter the product key and company name.
4. **Load OEM drivers.** The SSTK enables you to script the loading of OEM device drivers not supplied by the OS. You can specify whether the driver is to be loaded during the text mode (for a boot controller) or the graphical mode of Windows Setup.
5. **Install additional applications.** HP urges you to update the replica servers with the latest drivers and utilities and set it up as a managed system. This involves installing the Integrity Support Pack (ISP) on each server. The SSTK makes this easy. You can extend the unattended installation process to include the ISP.

Setting up the toolkit environment

Setting up the toolkit environment involves the following tasks:

1. **Set up the toolkit launch.** Setting up the toolkit launch involves editing a script and a configuration file that control the toolkit. The EFI loader launches the Linux boot loader, which executes a bootstrap script in the root file system. The bootstrap script invokes the main toolkit launch script, which uses a Windows setup script to trigger the OS installation. Edit the Linux boot loader configuration file and the Windows setup script. You do not need to modify other files unless you change the repository structure.
2. **Set up a boot mechanism.** Edit the boot configuration file to specify the boot device, the location of the scripts and OS files, and the username and password to access that location (for a network share). Create an image using the boot files. Then, either burn this image onto the boot device or set up the remote installation server that would support a network boot and OS installation.

Launching the server replication

Launching the server replication process involves the following tasks:

1. **Launch the configuration capture.** Boot the model server using the SSTK boot device. Select the **read hardware configuration** (`readhwconfig`) option when prompted. The script then runs utilities that capture and write the server configuration to the repository on the SSTK workstation. Edit these configuration files if needed.

2. **Launch the hardware configuration and software installation.** If the target system is cell-based you must create a hard partition for OS installation. Select the **Windows installation** (`win`) option when prompted. The script runs utilities that configure the target server hardware and then launches the software installation. After the process ends, verify that the OS is up and running, and specify system settings to complete the installation.

Customizing the toolkit environment

After you have gained some familiarity with the SSTK, you might find yourself eager to take advantage of its flexibility using your own scripts and customizations. HP suggests the following ways in which you can extend the capabilities of the toolkit environment.

- **Customize the configuration of target servers:** The sample scripts use the `data_files` directory for storing server configurations. This means that the same configuration is applied to all target servers represented by a platform. You can also select target servers using a different index and apply the configuration selectively. For example, you can specify a unique configuration for each server using its MAC address as the index.
- **Customize the Linux root file system:** The `rootfs.gz` file enables the server to boot to the toolkit environment with network connectivity. You can edit the network settings in the packaged `rootfs.gz` file to accommodate your deployment, but this may not be necessary.
- **Customize the Linux kernel:** We recommend that you use the kernel shipped with the toolkit because it has been tested on all supported Integrity servers. If you rebuild the kernel or obtain a kernel from another distribution, you will need to do extensive testing to verify that it runs with the hardware in HP Integrity servers.

Troubleshooting

SSTK uses a log mechanism to record errors and messages generated by the scripts. To generate the log, write permission must be enabled for `$REPOSITORY/data_files/log` in the SSTK repository.

SSTK creates a file, *`yyyymmdd-prodname-prodserial`*`.log`, where *`yyyymmdd`* is the current date (*`yyyy`* = year, *`mm`* = month, and *`dd`* = day), *`prodname`* is the machine product name, and *`prodserial`* is the product serial number.

For example, `20070615-server-Integrity-Virtual-Machine-VM00721000.log`

Log file format

For each Action command executed with an SSTK script, the following type of entry is recorded in the log file:

```
[14:43:03] Check repository contents and security
Command: check_repository
Return Value: 0
Standard output:
Standard error:
Security risk. Some folder(s) other than 'data_files' have write permission at the repository.
[14:43:05] done.

[14:43:05] Moving log to repository
Command: move_log2rep
Return Value: 0
Standard output:
Standard error:
[14:43:05] done.

[14:43:06] Copying toolkit and configuration files
Command: copy_toolkit
Return Value: 0
Standard output:
Standard error:
acu.ini not found.
hba-emulex.ini not found.
hba-qlogic.ini not found.
[14:43:07] done.
```

If the LOG_LEVEL variable in the sstk.conf file is set to 0 (zero), addition information is added to the log file as indicated in the following example:

```
============== lspci ==============
<output of the lspci command>

============== dmesg ==============
<output of the dmesg command>

============== hpbmc ==============
<output of the hpbmc utility>

============== dumppci ==============
<output of the dumppci utility>

============== hpia64info ==============
<output of the hpia64info utility>

============== /proc/partitions ==============
<output of the /proc/partitions file>

============== hwdisc ==============
<output of the hwdisc utility>
```

2 Creating a server profile

A server profile is like the DNA of all the servers in the deployment. It includes the server EFI environment variables, the I/O configuration, the OS and drivers, and installed applications. The server profile is stored in the repository. This chapter shows you how to create the server profile. Because you need to perform this task on the workstation using the SSTK, we begin by setting up this workspace.

Prepare the workstation

The workstation is the base of all SSTK operations. Before you can use it, you must make sure it is equipped with all the tools you need. If you are using a Windows workstation, ensure that you have a text editor that can save files in the UNIX file format or a DOS to UNIX file converter. HP urges you to install the 32-bit version of the Partition Manager tool on the workstation. Use this utility to create a partition on the model server and the target server. Partition Manager is available on the Smart Setup CD. If needed, install applications that can create a disk image and burn a bootable CD or DVD.

The URLs and products listed here are only suggestions. You can use any applications that can perform equivalent functions.

- The following text editors run on Windows and can save a file to the UNIX format.
  - **vim:** http://www.vim.org/
  - **crimson:** http://www.crimsoneditor.com/
  - **textpad:** http://www.textpad.com/
  - **dos2unix** converter: http://www.simtel.net/pub/pd/51438.html
- SSTK has been tested with the **WinImage** disk imaging application. You can obtain WinImage at http://www.winimage.com/
- SSTK has been tested with the following CD and DVD burning applications:
  - **Roxio Easy CD Creator:** http://www.roxio.com/
  - **Nero:** http://www.nero.com/

Display headless server console output at the workstation

In order to display the console output of a "headless" Linux Integrity server at a remote workstation, you must configure the Integrity server to use a particular console device. On Integrity servers running Linux, the console cannot display over the MP LAN as well as the local console. So if you intend to display the console output at the workstation via a telnet connection, you must first modify the Integrity server configuration to specify that the input, output, and error consoles use a single "Uart" device.

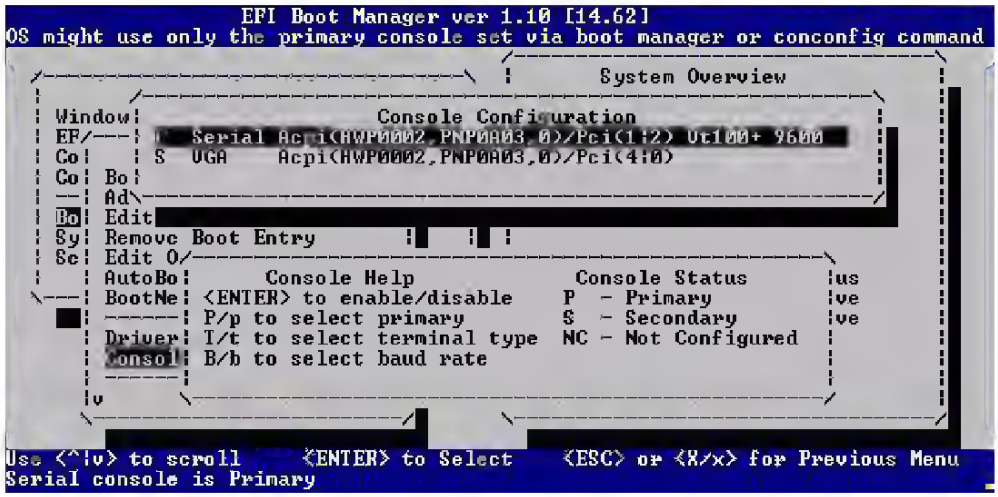
NOTE: The EFI Boot Manager may be used from the Enhanced interface (displayed with a grey background) or the Legacy interface (displayed with a black background).

At the Integrity server (the model server and each target server) using the Enhanced interface:

1. Select **EFI Boot Manager**->**Boot Configuration**->**Console configuration**. (Figure 2-1.)
2. Select **Input Console**. Select only one `Acpi(HWP0002,xxx)PCI ~~/Uart~~` device.
3. Select **Output Console**. Set output console to be the same as the input console device.

4. Select **Error Console**. Set error console to the same as the output console device.

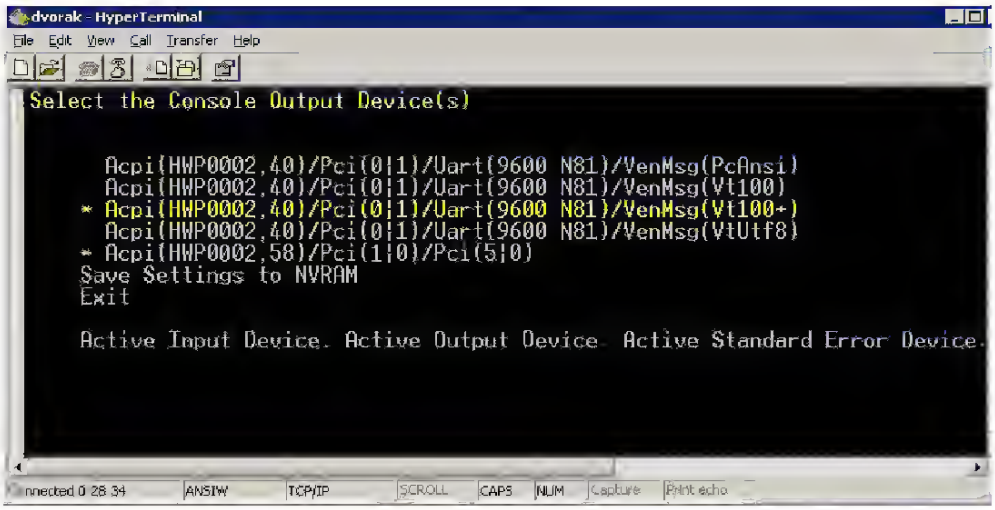
Figure 2-1 Console Configuration Window



At the Integrity server (the model server and each target server) using the Legacy interface:

1. Select **EFI Boot Manager->Boot Option Maintenance Menu.**
2. Select **Select Active Console Input Devices.** Select only one `Acpi(HWP—2,700)PCI ~~/Uart~~` device.
3. Select **Select Active Console Output Devices.** Set input console to be the same as the output console device. (Figure 2-2)
4. Select **Select Active Standard Error Devices.** Set error console to the same as the output console device.

Figure 2-2 Console Output Devices Window



NOTE: If multiple consoles are configured, the console output defaults to the local video console. The workstation is then unable to display the console output (or the Linux prompt). If the input, output, and error consoles do not match, the system firmware automatically matches the input console with output console.

On cell-based servers, select the serial device `Acpi(HWP0002,xxx)PCI ~~/Uart~~` as the primary input and output device, and the VGA device as the secondary input and output device.

Unpack the toolkit

The SSTK software package is an archive that is available one of the following file formats:

ZIP file	The `.zip` file is intended for deployment environments in which most workstations run Windows. You can extract files using a ZIP file utility (WinZip).
tar gzip file	The `.tgz` file is intended for deployment environments in which most workstations run Linux or HP-UX. You can expand and extract the archive using the `tar` command.

The archive expands into a file structure (Figure 2-3) that also serves as a skeleton for the server profile.

Figure 2-3 SSTK Archive

The following list shows the directory hierarchy and the files within those directories. The root of the directory contains this guide and a README file with late-breaking release information.

- The `boot_files` directory holds the toolkit environment, which you must copy onto the SSTK boot device. The `boot_files\efi\boot\` directory contains the following files:

  `bootia64.efi` `\efi\boot\bootia64.efi` is executed when an Itanium-based Integrity server boots. On cell-based systems it checks and sets the

	acpi flag to the appropriate state single-pci-domain or windows and then boots.
`bootia64.conf`	configuration file used to configure the boot loader and ACPI mode
`elilo.conf`	configuration file for the Linux boot loader
`elilo.efi`	Linux boot loader
`linux`	Linux kernel
`rootfs.gz`	Linux root file system in a compressed file

- The `data_files` directory is a placeholder that is used to store configuration files and SSTK log files.

`acu.ini`	The captured RAID configuration file. This file is created when the `readhwconfig` option is executed on a target server with a supported RAID controller. The `acu.ini` file is read when the `windows-setup.sh` script is executed. This file is not included in the sample file structure.
`conrep.dat`	The conrep EFI variables output file. This file is created when the `readhwconfig` option is executed.
`hba-qlogic.ini`	The `hpqlarep` configuration file, used to enable or disable the EFI boot setting on QLogic host bus adapters (HBAs). This file is also captured when the `readhwconfig` option is executed, and applied before installing Windows.
`hba-emulex.ini`	The `hplpgfg` configuration file, used to enable or disable the EFI boot setting on Emulex HBAs. This file is also captured when the `readhwconfig` option is executed, and applied before installing Windows.
`log`	This folder stores log files. It is created during SSTK execution.

- The `drivers` directory is the placeholder directory of OEM driver files. For example, to properly load the Intel NIC drivers, you must place the drivers in the following directory:

  `Network/intelNIC/windows/2003-enterprise/other-files/$OEM$/$1/PnPDrivers/Network/intelNIC/`

- The `software` directory is the placeholder of applications copied to the HP Service Partition or Windows tools that will be installed after the operating system is initially installed. Files placed under `software\EFI_TOOLS` will be automatically copied to the HP Service Partition.

NOTE: When installing Windows through the "win" option, ACU runs with –internal switch, which prevents overwriting Emulex or QLogic HBA configuration in a SAN (storage area network) environment.

- The `os` directory contains a sample unattended installation answer file, which you can customize for your deployments.
- The `scripts` directory contains the following shell scripts (typically, you can customize the first set of scripts for each option in the SSTK main menu).

  Deployment scripts:

`capture.sh`	Runs the `conrep`, `acu`, and `hpqlarep` utilities, capturing the hardware configuration and saving the results in the repository under the `data_files` directory.
`deploy.sh`	Replicates the captured information from the repository to the target system.
`windows-setup.sh`	Launches Windows installation by creating and formatting disk partitions, copying the operating system files to the MSDATA

	partition, copying the setup loader (`setupldr.efi`) application to the ESP partition, configuring the `winnt.sif` file, specifying the Windows Setup boot option, and rebooting the server.
`sstk.conf`	In this file, you can set the SSTK variables used by the scripts, such as the default action of SSTK, the disk where windows will be installed, the name of configuration files, and so on.

SSTK interface and auxiliary functions:

`main_toolkit_launch.sh`	Sets up the toolkit environment by creating a RAMdisk, copying scripts and utilities to the RAMdisk, running I/O hardware inventory, and prompting you to select deployment task.
`raid1-lsi.sh`	Contains the functions for the RAID configuration in LSI controllers.
`sstkw_functions`	Auxiliary functions for the scripts.

- The `utilities` directory contains the following files:

`CLMChecking`	A utility that performs memory optimization in cellular systems.
`allboards.xml`	A PCI device list file is used by the `hwquery` and `ifhw` utilities to convert PCI IDs found in hardware discovery into device names.
`cfggen`	A utility that is the RAID configuration utility for LSI controllers used in SSTK for applying RAID 1 in LSI 1068 controllers. The utility works in conjunction with `cpqacuxe`.
`conrep`	The configuration replication utility that reads EFI variables from the model server and records them in the `conrep.dat` file.
`conrep.xml`	A file that includes a list of EFI variables which determines what data to retrieve from the server. You can edit this file to control the data collection.
`cpqacuxe-X.YY-Z_ia64.tar.gz`	The array configuration utility operates in capture mode (reads the configuration of a storage array controller and records it in a configuration file) or in input mode (writes the configuration to the target server). This archive is extracted to `/sbin/cpqacuxe` and a structure under `/opt/compaq/cpqacuxe`.
`efibootmgr`	The EFI boot manager utility loads EFI applications (such as the OS boot loader) and EFI drivers. The EFI boot manager allows you to control the server's boot environment using variables stored in NVRAM.
`efivar`	The `efivar` utility is a generic tool for setting EFI variables from SSTK LinuxPE.
`gen-setupldr-options`	The `gen-setupldr-options` utility generates options for the setup loader, which launches the Windows installation process.
`hpia64info`	The `hpia64info` utility displays information about Management Processor and system firmware.
`hwdisc`	The `hardware discovery` utility creates an inventory of the server and writes this data to a file.
`hpqlarep`	The `hpqlarep` utility enables or disables the EFI boot setting on QLogic 2 GB or 4 GB HBAs.

`getuuid`	The `getuuid` utility retrieves the universally unique identifier (UUID) used for identifying a storage device or a single partition.
`hplpcfg`	The HP Emulex HBA configuration utility.
`hpqlarep`	The HP QLogic HBA configuration utility.
`hpshowdisk`	The `hpshowdisk` utility gathers information from storage devices, such as disk size, device node, controller type, target logical unit number (LUN), HBA world wide identification number (WWID), and target port WWN for disks in a SAN environment.
`hwquery`	The `hardware query` utility reads the file created by the hardware discovery utility and provides this data in a usable form.
`ifhw`	The `if hardware` utility makes user process decisions based on data from the file created by the hardware discovery utility.
`parted`	The `parted` utility creates, destroys, resizes, checks, and copies partitions.
`hpsetacpi`	The `hpsetacpi` utility sets the acpi mode for cellular systems.
`sstkencrypt`	The `sstkencrypt` utility hides the repository password, which is set in `elilo.conf`, over a Windows network. There is a Linux (ia64) and Windows version of this utility.

Setting permissions for repository access

To increase security, write permission must be restricted for directories in the SSTK repository. This also guarantees the integrity of files which should have read-only permission.

The repository structure has changed with SSTK 2.7, moving the read-only `drivers` and `software` directories to the root of the repository to enhance security and simplify configuring permissions on Windows. The new folder structure allows users to have read-write permissions for only the `data_files` directory. All other directories must be set to read-only. SSTK checks the user permissions after the repository is mounted and logs a warning message if permissions are not set correctly. Table 2-1 indicates appropriate permissions for each SSTK directory.

Table 2-1 SSTK Repository Permissions

Directory	Linux	Windows 2003 (ACLS) Basic View
`boot_files`	`r-x`	Read and execute
`data_files`	`rw`	Read, write
`os`	`r-x`	Read and execute
`scripts`	`r-x`	Read and execute
`software`	`r-x`	Read and execute
`drivers`	`r-x`	Read and execute
`utilities`	`r-x`	Read and execute

For additional information about sharing permissions and password configurations over a Windows Network, see the following resources:

- *How To Share Files and Folders over the Network in a Windows Server 2003 Domain Environment* at http://support.microsoft.com/kb/324267/EN-US
- *How to Configure File Sharing in Windows XP* at http://support.microsoft.com/kb/B304040/EN-US

CLM mode check

On cellular systems, the available system memory can be accessed using two different approaches: interleaved memory, meaning that the average memory latency is constant for all processes in the system; and non-interleaved memory, which provides lower memory latency for processes running in the same cell as the memory. Non-interleaved memory is also called cell local memory (CLM) or global non-interleaved (GNI) memory. Microsoft Windows has better performance when the memory is configured with maximum CLM.

During a scripted install, SSTK checks if the memory is correctly configured using the CLMChecking utility. If the configuration is not optimal, the following message appears:

```
For optimal performance of Windows in a partition, set the CLM to 100%
in all cells. Refer to the nPartition Guide for detailed instructions
on setting CLM using the Partiton Manager or the nPartition command
line interface. Press Ctrl+C for configuring now or wait for reboot.
```

NOTE: SSTK does not provide a tool to configure the memory.

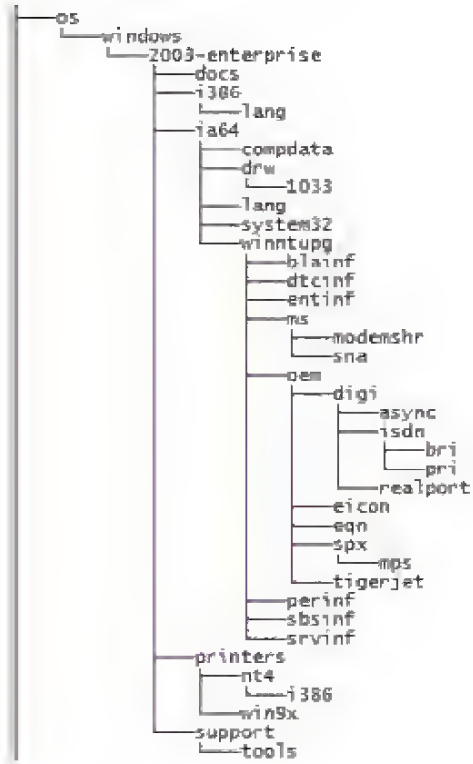
Upload the operating system installation files

To install an operating system on the target servers, the operating system installation files copied from the Microsoft RTM CDs to the repository must be accessible to the SSTK. The installation files of all supported operating system editions can be hosted in the repository, and scripts can be used to install the appropriate operating system to specific target servers. For example, you could upload the operating system files for the English and the Japanese versions of the SP1 and non-SP1 versions and install a particular operating system based on server selection criteria.

NOTE: The Integrity SSTK currently supports Microsoft Windows Server 2003, Enterprise Edition for Itanium-based systems in English and Japanese with and without SP1 or SP2.

Populate the `/os/windows/2003-enterprise/` directory with the unmodified contents of the operating system installation CD. If needed, you can create additional directories for other editions or versions of the operating system as a peer to this directory. Figure 2-4 shows the resulting directory structure.

Figure 2-4 /os/windows/2003-enterprise/ Directory



In case you want to configure your repository to enable other Windows operating system versions, you must do the following:

1. Populate the `/os/windows/` directory as previously described.
2. Populate the `/drivers/` directory for all desired versions.

The resulting directory structure is as follows:

```
|
+---os
|   \---windows
|       |---2003-enterprise
|       \---2003-enterprise-sp2
|
+---drivers
|   \---storage
|       |---saP600
|       |   \---windows
|       |       |---2003-enterprise
|       |       \---2003-enterprise-sp2
|       \---sa6400
|           \---windows
|               |---2003-enterprise
|               \---2003-enterprise-sp2
|
```

Set up an unattended operating system installation

The Windows directory contains a file to drive an unattended Windows installation. This file, which Microsoft refers to as an "answer file", is called `unattended.txt`. Even if you maintain multiple Windows versions, the same `unattended.txt` file will be used for each version. For more information about the answer file, go to the Microsoft Windows Server 2003 TechCenter Web site at http://technet.microsoft.com/en-us/windowsserver/default.aspx

The SSTK provides a sample `unattended.txt` file that you can edit. You must enter the product key under the `Unattended` section.

```
[Data]
MsdosInitiated = 1
winntupgrade="no"

[Unattended]
DriverSigningPolicy = Ignore
NTUpgrade = No
Win9xUpgrade = No
OemSkipEula = Yes
Unattendmode = FullUnattended
OemPreinstall = No
TargetPath = *
Filesystem = ConvertNTFS
WaitForReboot = No
FactoryMode = Yes
ExtendOemPartition = 1

# The OemPnPDriversPath variable will be updated by SSTK
OemPnPDriversPath =

[UserData]
FullName = "Company"
OrgName = "Company"
ComputerName = ComputerName
ProductKey=XXXXX-XXXXX-XXXXX-XXXXX-XXXXX

[GuiUnattended]
AdminPassword = "admin"
OEMSkipRegional = 1
OEMSkipWelcome = 1
AutoLogon=Yes
AutoLogonCount=1
TimeZone = 004

[LicenseFilePrintData]
AutoMode = "PerServer"
```

```
AutoUsers = "5"

[Display]
BitsPerPel = 16
XResolution = 800
YResolution = 600
VRefresh = 70

[Branding]
BrandIEUsingUnattended = No
IEBrandingFile = install.ins

[Networking]

[WindowsFirewall.Standard]
Services = RemoteDesktop

[WindowsFirewall.RemoteDesktop]
Type = 2

[TerminalServices]
AllowConnections = 1

[Identification]
JoinWorkgroup = Workgroup

[NetOptionalComponents]
SNMP=1
WBEMSNMP=1
SimpTCP=1

[SNMP]
Contact_Name="xxx"
Location="xxx"
Accept_CommunityName=PUBLIC:Read_Only
Accept_CommunityName=xxx:Read_Create
Traps=xxx

[Identification]
JoinWorkgroup = Workgroup

[GuiRunOnce]
#This line is to install all software from REPOSITORY/software/windows/
"%SystemDrive%\Installers\PostInstall.cmd"
```

Accessing the server after a Windows installation

After installation the Windows firewall is enabled by default. You can use the following commands to disable the firewall.

```
[WindowsFirewall]
Profiles = WindowsFirewall.TurnOffFirewall
[WindowsFirewall.TurnOffFirewall]
Mode = 0
```

If the firewall is enabled, you can use the following commands to enable the remote desktop through the firewall (as in the previous `unattended.txt` file.

```
[WindowsFirewall.Standard]
Services = RemoteDesktop
[WindowsFirewall.RemoteDesktop]
Type = 2
[TerminalServices]
AllowConnections = 1
```

Instolling tools from the Integrity Support Pock

To install Integrity Support Pack tools, copy the executable file for each tool you want to install into the appropriate directory, `\software\windows\ISP`, in the repository. The Support Pack tools are located on the HP SmartSetup CD in the `\contents\supportpack\isp` directory. Edit the `PostInstall.cmd` file located at `\software\windows\`, adding command line to install each tool.

After a Windows installation, `Postinstall.cmd` is executed and each utility listed on this file and located in the repository directory is copied to the `%SystemDrive%\Installers\` directory and executed according to the command line specified.

For example, to execute the `setupc.exe` utility, copy the executable file to the `\software\windows\ISP` directory in the repository. Add the following line to the `PostInstall.cmd` file:

```
%SystemDrive%\Installers\ISP\setupc.exe /use-latest /reboot
```

Copying softwore to the HPSP portition

The HPSP partition is created during SSTK deployment. This partition has 400 MB of free space available. To copy specific software to the HPSP partition, copy it to `\software\EFI_TOOLS\`. This partition is formatted with the FAT32 file system, so it is accessible from EFI shell and from the operating system after deployment.

Upload the device driver files

Some OEM drivers are not on the operating system RTM CD. To install these additional drivers, you must upload the driver files into the SSTK repository structure and integrate their installation into the Windows setup process. This section details how to upload the driver files into the correct structure, and the next section details how to integrate the files into Windows setup.

1. In the `/drivers/` directory at the root of the SSTK repository, create a subdirectory for each attribute of the driver. Sample directories have already been created as examples. Create the subdirectory in the following order:
   - The class of hardware (for example, storage or network)
   - The device name (saP600, sa6402, or any other free-form name)
   - Operating system family (Windows)
   - Operating system edition (2003-enterprise)

   For example, the partial path to the driver for the Smart Array P600 boot controller is as follows: `/drivers/storage/saP600/windows/2003-enterprise/`.

NOTE: For each operating system edition or version (such as Windows Server 2003 Enterprise Edition and Windows Server 2003 Enterprise Edition SP2), you must create a corresponding driver directory. For an example, see the sample directory tree in the "Upload the operating system installation files" (page 30) section.

The path must also indicate whether the driver is to be loaded in the first phase ("text mode") or the second phase ("graphical mode") of the Windows setup process.

- In text mode, only those drivers needed to boot the system are loaded. For example, if Windows is to be installed on a disk in the array connected to the P600 controller, the driver for this controller must be loaded before Windows can be installed. To enable the use of this mode, create the `boot-files/$OEM$/TEXTMODE` directory under `/drivers/storage/saP600/windows/2003-enterprise/`.
- In graphical mode, drivers for all the devices, including boot devices, are loaded. To support the loading of all OEM drivers, create the `other-files/$OEM$/$1` directory under `/drivers/storage/saP600/windows/2003-enterprise/`.

2. Copy driver files into these directories. If you copied a driver to the `TEXTMODE` directory, you should also copy it to the `other-files/$OEM$/$1` directory.

The final repository may resemble the following structure (operating system files were removed for the sake of brevity):

```
|   readme.txt
|   SSTK_DeploymentGuide.pdf
|
+---boot_files
|   \---efi
|       \---boot
|               bootia64.efi
|               bootia64.conf
|               elilo.conf
|               elilo.efi
|               linux
|               rootfs.gz
|
+---data_files
|   |   acu.ini
|   |   hba-emulex.ini
|   |   hba-qlogic.ini
|   |   conrep.dat
|   |   log/
|   |
+---drivers
|   +---Network
|   |   +---AD3xx
|   |   |   \---windows
|   |   |       \---2003-enterprise
|   |   |           |---other-files
|   |   |           |   \---$OEM$
|   |   |           |       \---$1
|   |   |           |           \---PnPDrivers
|   |   |           |               \---Network
|   |   |           |                   \---AD3xxA
|   |   |           |                           NicCo.dll
|   |   |           |                           NicEtCo.dll
|   |   |           |                           NicEtCoE.dll
|   |   |           |                           NicInstE.dll
|   |   |           |                           PROUnstl.exe
|   |   |           |                           e1000msg.dll
|   |   |           |                           e1e5164.cat
|   |   |           |                           e1e5164.din
|   |   |           |                           e1e5164.inf
|   |   |           |                           e1e5164.inf
|   |   +---broadcom
|   |   |   \---windows
|   |   |    \---2003-enterprise
|   |   |           |
|   |   |           \---other-files
|   |   |               \---$OEM$
|   |   |                   \---$1
|   |   |                       \---PnPDrivers
|   |   |                           \---Network
|   |   |                               \---broadcom
|   |   |                                       b57xp64.cat
|   |   |                                       b57xp64.inf
|   |   |                                       b57xp64.sys
|   |   |
|   |   +---intelNIC
|   |   |   \---windows
|   |   |       \---2003-enterprise
|   |   |           |
```

```
|   |   |           \---other-files
|   |   |               \---$OEM$
|   |   |                   \---$1
|   |   |                       \---PnPDrivers
|   |   |                           \---Network
|   |   |                               \---intelNIC
|   |   |                                       e1000645.cat
|   |   |                                       e1000645.din
|   |   |                                       e1000645.inf
|   |   |                                       e1000645.sys
|   |   |                                       e1000msg.dll
|   |   |                                       EtCoIn64.dll
|   |   |                                       generic.dll
|   |   |                                       install.xml
|   |   |                                       intlnc64.dll
|   |   |                                       Ncs2Setp.dll
|   |   |                                       PROUnstl.exe
|   |   |
|   |   \---neterion
|   |       \---windows
|   |           \---2003-enterprise
|   |               |
|   |               \---other-files
|   |                   \---$OEM$
|   |                       \---$1
|   |                           \---PnPDrivers
|   |                               \---Network
|   |                                   \---neterion
|   |                                           netxena.cat
|   |                                           netxena.inf
|   |                                           xena64.sys
|   |
|   +---Others
|   \---storage
|       +---emulex
|       |   \---windows
|       |       \---2003-enterprise
|       |           +---boot-files
|       |           |   \---$OEM$
|       |           |       \---TEXTMODE
|       |           |               lpscsi
|       |           |               lpsimdev.cat
|       |           |               lpsimdev.inf
|       |           |               lpxftr.sys
|       |           |               lpxnds.cat
|       |           |               lpxnds.dll
|       |           |               lpxnds.sys
|       |           |               oemsetup.inf
|       |           |               txtsetup.oem
|       |           |
|       |           \---other-files
|       |               \---$OEM$
|       |                   \---$1
|       |                       \---PnPDrivers
|       |                           \---Storage
|       |                               \---emulex
|       |                                       lpscsi
|       |                                       lpsimdev.cat
|       |                                       lpsimdev.inf
|       |                                       lpxftr.sys
|       |                                       lpxnds.cat
|       |                                       lpxnds.dll
|       |                                       lpxnds.sys
|       |                                       oemsetup.inf
|       |                                       txtsetup.oem
|       |
```

```
|       |
|       +---qlogic4q
|       |   \---windows
|       |       \---2003-enterprise
|       |           +---boot-files
|       |           |   \---$OEM$
|       |           |       \---TEXTMODE
|       |           |               ql2300.cat
|       |           |               ql2300.sys
|       |           |               ql2x00.inf
|       |           |               txtsetup.oem
|       |           |
|       |           \---other-files
|       |               \---$OEM$
|       |                   \---$1
|       |                       \---PnPDrivers
|       |                           \---Storage
|       |                               \---qlogic4g
|       |                                       ql2300.cat
|       |                                       ql2300.sys
|       |                                       ql2x00.inf
|       |                                       ql2xhai2.dll
|       |                                       qlogic
|       |                                       qlogic
|       |                                       txtsetup.oem
|       |
|       +---sa6400
|       |   \---windows
|       |       \---2003-enterprise
|       |           |
|       |           +---boot-files
|       |           |   \---$OEM$
|       |           |       \---TEXTMODE
|       |           |               cpqciss.dll
|       |           |               cpqcisse.cat
|       |           |               cpqcisse.inf
|       |           |               cpqcisse.pdb
|       |           |               cpqcisse.sys
|       |           |               cpqcissm.cat
|       |           |               cpqcissm.dll
|       |           |               cpqcissm.inf
|       |           |               cpqcissm.pdb
|       |           |               cpqcissm.sys
|       |           |               install.xml
|       |           |               txtsetup.oem
|       |           |
|       |           \---other-files
|       |               \---$OEM$
|       |                   +---$$
|       |                   |   \---system32
|       |                   |           cpqcissm.dll0000755
|       |                   |
|       |                   \---$1
|       |                       \---PnPDrivers
|       |                           \---Storage
|       |                               \---sa6400
|       |                                       cpqciss.dll
|       |                                       cpqcisse.cat
|       |                                       cpqcisse.inf
|       |                                       cpqcisse.pdb
|       |                                       cpqcisse.sys
|       |                                       cpqcissm.cat
|       |                                       cpqcissm.dll
|       |                                       cpqcissm.inf
|       |                                       cpqcissm.pdb
```

```
|       |                                                cpqcissm.sys
|       |                                                install.xml
|       |                                                txtsetup.oem
|       |
|       \---saP600
|           \---windows
|               \---2003-enterprise
|                   |
|                   +---boot-files
|                   |   \---$OEM$
|                   |       \---TEXTMODE
|                   |               generic.dll
|                   |               hpcisss2.cat
|                   |               hpcisss2.sys
|                   |               hpcissx2.inf
|                   |               txtsetup.oem
|                   |
|                   \---other-files
|                       \---$OEM$
|                           \---$1
|                               \---PnPDrivers
|                                   \---Storage
|                                       \---saP600
|                                               generic.dll
|                                               hpcisss2.cat
|                                               hpcisss2.sys
|                                               hpcissx2.inf
|                                               txtsetup.oem
|
+---software
|   +---EFI_TOOLS
|   |   +---networking
|   |   |       ftp.efi
|   |   |       ifconfig.efi
|   |   |       ping.efi
|   |   |       route.efi
|   |   |       SETUPLDR.EFI
|   |   |       tcpipv4.efi
|   |   |
|   |   \---toolkit
|   |       |   Readme.txt
|   |       |   version.txt
|   |       |
|   |       \---PYTHON1.5
|   \---windows
|       |   PostInstall.cmd
|       |
|       \---ISP
|               cp006523.exe
|               cp006535.exe
|               cp006567.exe
|               cp006569.exe
|               cp006573.exe
|               cp006578.exe
|               cp006579.exe
|               cp006607.exe
|               cp006691.exe
|               cp006834.exe
|
+---os
|   |
|   \---windows
|       \---2003-enterprise
+---scripts
|       capture.sh
```

```
|       deploy.sh
|       main_toolkit_launch.sh
|       raid1-1si.sh
|       sstk.conf
|       sstkw_functions
|       windows-setup.sh
|
\---utilities
    |   CLMChecking
    |   allboards.xml
    |   cfggen
    |   conrep
    |   conrep.xml
    |   cpqacuxe-X.YY-Z_ia64.tar.gz
    |   efibootmgr
    |   efivar
    |   gen-setupldr-options
    |   getuuid
    |   hplpcfg
    |   hpqlarep
    |   hpshowdisk
    |   hwdisc
    |   hwquery
    |   ifhw
    |   parted
    |   hpsetacpi
    |   sstkencrypt
    |   windows
    |       \---sstkencrypt.exe
```

Set up driver installation

To install an OEM driver, the SSTK must copy the driver files from the repository to the target server and update the unattended file which instructs Windows setup to install the driver.

To take advantage of these capabilities, you must upload the driver files into the repository (as described in "Upload the operating system installation files" (page 30)). This section describes how the `windows-setup.sh` script copies available drivers from the OEM folder, the identification of the boot controller type, and items that are added to the unattended file. As previously detailed, the PnP drivers are installed during the graphical mode of the Windows Setup and copied by the `copy_pnp_drivers()` function in the `windows-setup.sh` script.

```
copy_pnp_drivers()
{
    # Copy PnPDrivers driver files to windows installation path

    # Destination folder

    local targdir="${MSDATA}/\$win_nt\$.~ls/\$OEM\$/"

    mkdir -p $targdir || return 1

    find ${SOURCE}/drivers/*/*/windows/${OS_VERSION}/other-files/\$OEM\$ \\
    -mindepth 1 -maxdepth 1 -exec cp -R {} $targdir \; || return 1

    return 0
}
```

The `$MSDATA`, `$SOURCE` (set in `sstk_defines`), and `$OS_VERSION` (set in `sstk.conf`) SSTK variables refer to the Windows MSDATA partition at the target server, the repository path, and the operating system edition, respectively.

SSTK automatically sets the `OemPnPDriversPath` in the unattended file according to the driver repository structure. During Windows text mode setup, only the OEM driver for the boot

controller must be loaded. This is specified in the `[MassStorageDrivers]` and `[OEMBootFiles]` sections in the unattended file.

When you define the target disk where Windows will be installed (either setting the variable `$DISK` in `sstk.conf` or through the SSTK menu) the `hpshowdisk` utility automatically retrieves the controller type, which can be difficult to obtain if the system has a large number of disks and controller types.

In the `windows-setup.sh` script, the function `copy_textmode_drivers()` copies the correspondent OEM driver for the boot controller to the folder `\${MSDATA}\$win_nt$.~ls\$OEM$\TEXTMODE\`. In addition, you must be sure that the `windows-setup.sh` script handles the controller type being installed. If necessary, you can add new controller models using the `get_controller()` function. The `hpshowdisk` utility returns a unique PCI ID which specifies the controller type.

NOTE: You can obtain the controller PCI ID from the `*.INF` file of the OEM driver.

The `get_controller()` function returns the folder name where the driver files are stored. If the controller type is not set within this function, the value `retail` is returned. This value directs the setup utility to install the driver included with Windows Server 2003.

In most cases, it is sufficient to use the vendor ID to match the controller type. Also, the `$CONTROLLER` variable (set in the `sstk.conf` file) can be used to bypass this functionality and force the folder name where the OEM driver files are stored.

```
get_controller()
{
    local ids=($($UTILITIES/hpshowdisk -i $selected_disk_number -p | tr ':' ' '))

    if [ ${ids[0]} = "1000" ]; then
        [ ${ids[1]} = "0050" ] && echo "lsi1068" && return
        [ ${ids[1]} = "0054" ] && echo "lsi1068" && return
        [ ${ids[1]} = "0056" ] && echo "lsi1068" && return
        [ ${ids[1]} = "0058" ] && echo "lsi1068" && return
        echo "retail"

        return
    fi
    [ ${ids[0]} = "0e11" ] && echo "sa6400" && return
    [ ${ids[0]} = "103c" ] && echo "saP600" && return
    [ ${ids[0]} = "10df" ] && echo "emulex" && return
    [ ${ids[0]} = "1077" ] && echo "qlogic4g" && return

    echo "retail"
    write_warning "Unknown storage device driver, using retail driver."

    return
}
```

If an OEM diver will be used, the `update_unattended_file()` function automatically appends the sections `[MassStorageDrivers]` and `[OEMBootFiles]` to the driver. The information required to update the unattend file is obtained from the `txtsetup.oem` file of the controller OEM driver. For example, for the Smart Array 6400 controller, SSTK appends the following data to the unattend file:

```
[MassStorageDrivers]
"Smart Array 6i, 641, 642, 6400, 6400EM Controllers" = "OEM"

[OEMBootFiles]
cpqcisse.cat
cpqcisse.inf
cpqcisse.sys
cpqcissm.cat
cpqcissm.inf
cpqcissm.sys
```

3 Setting up the toolkit environment

Setting up the toolkit environment involves editing scripts that control the boot and operation of the toolkit. The EFI loader launches the Linux boot loader, which in turn executes a bootstrap script in the root file system. The bootstrap script invokes the main toolkit launch script, which uses a Windows setup script to trigger the operating system installation. This chapter provides guidance on editing these files.

Edit sstk.conf

The SSTK variables are grouped in the `sstk.conf` file. This section describes each variable and the default values.

`DEFAULT_MAIN_MENU_OPTION=""` Defines the default action of SSTK. The default value is empty, and any of the following values can be entered:

`win`	Performs a Windows installation
`readhwconfig`	Reads hardware configuration variables
`bash`	Enters at the bash prompt
`lsi`	Configure RAID 1 on systems having LSI 1068 controllers

`INTERACTIVE="no"` Prompts you to confirm certain actions, such as overwriting files. The value can be set to "yes" or "no," and the default value is "no."

`WAIT_TIMEOUT=0` Used in interactive mode to time out the message prompts. The value can be set to an integer number of seconds.

`DISK=""` Sets the target device where Windows is going to be installed, for example, `/dev/sda`. The default value is empty so that a disk menu appears.

`CONTROLLER=""` Sets the folder name of the OEM driver in the driver repository, for example, `sa6400`. The default value is empty so that the controller is set automatically by SSTK. If this value is set to "`retail`," Windows setup installs a driver included with Windows 2003.

`OS_VERSION="2003-enterprise"` Sets the operating system edition of Windows to be installed. The default value is "2003-enterprise." For example, if the directories `$OSDIR/windows/2003-enterprise` and `$OSDIR/windows/2003-enterprise-jp` contain the Windows installation files and you want to install Windows Server 2003 Enterprise Edition Japanese, the `OS_VERSION` variable must be set to `OS_VERSION="2003-enterprise-jp"`.

The following variables set the name of the configuration files used throughout the scripts. The default values are as follows:

- `ACU_FILE="acu.ini"`
- `CONREP_CONF="conrep.xml"`
- `CONREP_FILE="conrep.dat"`
- `HBAQLOGIC_FILE="hba-qlogic.ini"`
- `HBAEMULEX_FILE="hba-emulex.ini"`
- `LOG_LEVEL=0`: Sets the log level. The default value is "`0`." Set "`1`" for normal log or "`0`" for extended log.

Edit bootia64.conf

The `bootia64.conf` file configures how the SSTK boot loader (`bootia64.efi`) loads the preboot environment. The `bootia64.conf` file distributed with SSTK is configured to load the Linux PE image and typically does not need to be modified.

```
loader_name = elilo.efi
acpi_mode = linux
#OS to ACPI mode map (OS detection feature).
os_acpi_map = windows;\EFI\Microsoft\Boot\bootmgfw.efi
os_acpi_map = windows;\EFI\Microsoft\WINNT50\ia64ldr.efi
os_acpi_map = windows;\setupldr.efi
```

`loader_name` Specifies the name of the boot loader. `bootia64.efi` runs, performs any required setup, and executes this file.

`acpi_mode` Specifies the proper ACPI mode in which to run the preboot environment. `bootia64.efi` sets the ACPI mode before executing the boot loader. Setting the ACPI mode to Linux automatically reverts to single-pci-domain or default depending on the server chipset. This line does not affect non-cellular servers.

`os_acpi_map` After setting the ACPI mode to the appropriate value to boot the preboot image, `bootia64.efi` reboots to update the ACPI table, and then runs only the configured boot loader. Before executing the boot loader, `bootia64.efi` sets the ACPI mode back to the operating system installed in the machine. This means that, after leaving the preboot environment, the server can run the pre-existing operating system. The `os_acpi_map` associates the operating system boot loader with the ACPI mode. In the distributed file there are three entries associating the Windows 2008 boot loader, Windows 2003 boot loader, and Windows 2003 setup loader to the Windows ACPI mode.

Edit elilo.conf

The SSTK environment starts when the EFI boot loader launches the LinuxPE boot loader (`bootia64.efi`). The Linux boot loader reads the `elilo.conf` configuration file and loads the kernel and root file system based on these configuration parameters. This is no different than booting up to the Linux operating system. But you can use the `elilo.conf` file to do much more than that.

The SSTK uses the `elilo.conf` file to run the bootstrap script in the root file system, mounts the repository file system, and launches the `main_toolkit_launch.sh` script located in the repository. Then, the `main_toolkit_launch.sh` script prompts you to select one of three tasks: start a bash shell, read the hardware configuration of the server, or write to the hardware configuration and install the operating system, drivers, and applications.

The `elilo.conf` configuration file is an ASCII (not a UNICODE) file that controls the behavior of the Linux boot loader. The config file follows the LILO/x86 syntax. The configuration options are divided in three groups:

- The **global options**, which are listed first, affect the behavior of ELILO and apply to all images. It is possible to have an empty list of global options.

Table 3-1 Global options

Global Option	Description
`default=value`	Name the default image to boot. If not defined, ELILO will boot the first defined image.
`timeout=number`	The number of 10th of seconds to wait while in interactive mode before auto booting default kernel. Default is infinity
`delay=number`	The number of 10th of seconds to wait before auto booting when not in interactive mode. Default is 0
`prompt`	Force interactive mode

Table 3-1 Global options *(continued)*

Globol Option	Description
`verbose=number`	Set level of verbosity [0-5] Default is 0 (no verbose).
`root=filename`	Set global root filesystem for Linux/ia64.
`read-only`	Force root filesystem to be mounted read-only
`append=string`	Append a string of options to kernel command line
`initrd=filename`	Name of `initrd` file
`image=filename`	Define a new image.
`chooser=name`	Specify kernel chooser to use: `simple` or `textmenu`
`message=filename`	Printed on the main screen if supported by the chooser
`fX=filename`	Some choosers may take advantage of this option to display the content of a file when a certain function key X is pressed. *X* can vary from 1-12 to cover function keys F1 to F12
`noedd30`	Do not force the EDD30 EFI variable to TRUE when FALSE. In other words, do not force the EDD30 mode if not set
`quiet`	Limit the onscreen output messages during the preboot image loading process

- The **image options** are specific to a particular kernel image. Each kernel image must be identified with a logical name called a label. At least one image MUST be defined.

Table 3-2 Image options

Imoge Option	Description
`root=filename`	Set root filesystem for kernel
`read-only`	Force root filesystem to be mounted read-only
`append=string`	Append a string of options to kernel command line
`initrd=filename`	Name of `initrd` file
`label=string`	Logical name of image (used in interactive mode)
`description=string`	One line text description of the image

- The **SSTK command line options** for the linux kernel, which are specified as a string of values to the image "append" option, bring up the toolkit environment.

Table 3-3 SSTK command line options

SSTK Option	Description
`ssstk_init=filename`	Name of the script on the root file system that initializes the SSTK environment by copying the repository into the ramdisk workspace. This script is named bootstrap This does not need to change.
`ssstk_mount_type=string`	File system type of your repository If on a linux workstation (or repository server), specify `nfs`. If the repository is on Windows, specify `smbfs`.
`ssstk_mount=filename`	Network share or device that hosts the repository: nfs `[hostname|IP]:/directory` smbfs `//[hostname|IP]/directory` USB,CD,DVD `/dev/devicename`
`ssstk_mount_options=string`	`username=user` `password=mypass` (for SMB/CIFS repositories only)

Table 3-3 SSTK command line options *(continued)*

SSTK Option	Description
`ssstk_encryptpass=`*`encryptpass`*	Encrypted password created by the `sstkencrypt` tool for SMB/CIFS repositories over a network for security. This option overwrites the password string in `sstk_mount_option`. **NOTE:** The password length is limited to a maximum of 16 characters.
`ssstk_script=`*`filename`*	Name of main launch script
`ssstk_action=`*`string`*	Default response to the toolkit launch script prompt (win, readhwconfig or bash)
`idebus=`*`xx`*	Reports the VESA/PCI bus speed in MHz to the IDE driver, where *xx* is between 20 and 66 inclusive. It is used when tuning chipset PIO modes. For a PCI bus, 25 is used for P75 systems, 30 is used for P90,P120,P180 systems, and 33 is used for P100,P133,P166 systems. If in doubt, use `idebus=33` for PCI.

Edit the sample `elilo.conf` file included in the SSTK software package to match your deployment. Below is an annotated example of a working `elilo.conf` file:

- The prompt forces interactive mode and the timeout waits for 30 seconds.

  ```
  prompttimeout=300
  ```

- The first (default) entry boots the server to Linux and opens a console (`label=linux`). To change the default so that the server automatically runs the read hardware configuration, move all the commands in second bullet below (`label=readhw-windows`), to the first entry in the `elilo.conf` file.

  ```
  image=linux
  label=linux
  initrd=rootfs.gz
  root=/dev/ram
  append="rw ramdisk_size=524288 idebus=33"
  ```

- The following entry executes the main toolkit launch script by getting over a Windows network using the administrator user name and password. The script then reads the server hardware configuration and writes it to the repository.

  ```
  image=linux
  label=readhw-windows
  initrd=rootfs.gz
  root=/dev/ram
  append="rw ramdisk_size=524288 ssstk_init=/bootstrap
  ssstk_mount_type=smbfs ssstk_mount=//192.168.39.23/ssst13
  ssstk_script=scripts/main_toolkit_launch.sh
  ssstk_mount_options=username=Administrator,password=repository
  ssstk_action=readhwconfig"
  ```

- The following entry executes the main toolkit launch script by getting over a Windows network using the administrator user name and an encrypted password. The script then executes the `main_toolkit_launch.sh` utility.

  Use the `sstkencrypt` utility (under Windows or Linux) to encrypt the repository password. For example, from a Windows command prompt, run:

  ```
  C:\ssst13\utilities\windows>sstkencrypt.exe
  Type the repository password:
  repository
  Set the 'encryptpass' parameter in SSTK elilo.conf to:
  267727232c7e2f21292f

  image=linux
  label=smb encrypted
  initrd=rootfs.gz
  root=/dev/ram
  append="rw ramdisk_size=524288 idebus=33 quiet ssstk_init=/bootstrap ssstk_mount_type=smbfs
  ssstk_mount=//192.168.39.23/ssst13 ssstk_script=scripts/main_toolkit_launch.sh
  ssstk_mount_options=username=Administrator ssstk_encryptpass=267727232c7e2f21292f"
  ```

NOTE: The repository password can have up to 16 characters, including only uppercase letters, lowercase letters, and digits. No special characters are supported. Special characters include: **!, @, #, $, %, ¨, &, *, (,), ', ", _, +, =, -, {, }, [, ", á, à, â, ã, é, è, ê, í, ì, î, ó, ò, ô, õ, ú, ù.**

- The following entry gets the main script from a repository located on an NFS share.

```
image=linux
label=nfs
initrd=rootfs.gz
root=/dev/ram
append="rw ramdisk_size=524288 ssstk_init=/bootstrap
ssstk_mount_type=nfs ssstk_mount=repositoryserver:/nfsshare
ssstk_script=scripts/main_toolkit_launch.sh"
```

- The following entry mounts the repository located on a DVD, executes the main script with the directive to configure the hardware and install Windows.

```
image=linux
label=install-DVD
initrd=rootfs.gz
root=/dev/ram
append="rw ramdisk_size=524288 ssstk_init=/bootstrap
ssstk_mount=/dev/cdrom
ssstk_script=scripts/main_toolkit_launch.sh ssstk_action=win"
```

Review main_toolkit_launch.sh

The `main_toolkit_launch.sh` script sets up the toolkit environment, prompts you to read the hardware configuration or install the operating system, and responds to your input. Setting up the environment involves copying the repository to the ramdisk, making sure the utilities are executable, creating an inventory of the hardware (hardware discovery), and loading the OEM device drivers.

The script obtains your input by reading the *`ssstk_action`* variable in the `elilo.conf` file or your choice in the toolkit menu. You can select `win`, `readhwconfig`, `bash`, or `lsi`.

- When you select `win`, the script invokes the `windows-setup.sh` script to launch the Windows installation:

```
if [ ! -e "$RAMDRIVE/scripts/windows-setup.sh" ]
then
echo "Unable to start Windows setup. Script not found"
exit 1
fi
```

  If you select `win`, the `deploy.sh` script executes before `windows-setup.sh` and applies the configurations contained in `conrep.dat` (conrep), `acu.ini` (cpqacuxe), `hba-qlogic.ini` (hpqlarep), and `hba-emulex.ini` (hplpcfg).

- When you select `readhwconfig`, the following tasks are executed in the `capture.sh` script:

  1. Runs cpqacuxe to get the configuration of the storage array controller. The script writes this data to the `acu.ini` file.

     ```
     /usr/sbin/cpqacuxe -c "${CONFIGDIR}/acu.ini"
     ```

  2. Executes `conrep` to get the EFI environment variables from the server configuration. The script writes this data to the `conrep.dat` file.

     ```
     ./conrep --read -f ${CONFIGDIR}/conrep.dat
     ```

  3. Run the `hpqlarep` utility to enable or disable the EFI boot setting on QLogic 2 GB or 4 GB HBAs. The script writes this data to the `hba.ini` file:

```
$UTILITIES/hpqlarep /S "${CONFIGDIR}/hba.ini
```

4. Run the `hplpcfg` utility to enable or disable the EFI boot setting on HP Emulex HBAs. The script writes this data to the `hba-emulex.ini` file.

   ```
   $UTILITIES/hplpcfg /le "${CONFIGDIR}/hba-emulex.ini
   ```

- When you select `lsi`, the `raid1-lsi.sh` script runs to create RAID1 to the LSI 1068 controller. This option is only available if there is a 1068 controller in the system. The disk connected to the controller is converted to a single disk using RAID1.

If you maintain the repository structure suggested in this guide, you will not have to make any changes in this script. If you set the sstk_action variable in the `elilo.conf` file, you will not need to interact with the script (it will run silently) to execute the option you select (`readhwconfig` or `win`).

You can edit the `DEFAULT_MAIN_MENU_OPTION`, `INTERACTIVE`, and `WAIT_TIMEOUT` variables in the `sstk.conf` file as needed. For information, see "Edit sstk.conf" (page 41)

Edit windows-setup.sh

The `windows-setup.sh` script starts the Windows installation process after it partitions the hard disk and copies operating system and OEM files into the appropriate partitions.

The script performs the following operations:

1. The script creates four partitions: EFI System Partition (ESP), HP Service Partition (HPSP), Microsoft Reserved Partition (MSR), and the Microsoft Data (MSDATA) partition.
2. Formats these partitions.
3. Copies the operating system files from the repository to the MSDATA partition.
4. Copies the Windows setup loader (`setupldr.efi`) to the ESP partition.
5. Copies the `unattended.txt` file to the `winnt.sif` file on the MSDATA partition.
6. Creates a new boot option in the EFI boot manager which starts the Windows Setup process. This option is called Windows Setup. On some cellular systems, the `windows-setup.sh script` must set the ACPI mode to Windows in EFI mode, causing the system to reboot and boot again to the SSTK boot entry. If the SSTK launcher (`bootia64.efi`) was run from the EFI SHELL, after reboot it returns to the EFI SHELL, making it necessary to call the launcher manually again to set the EFI mode to Windows. After the mode is set, the machine reboots again and boots to the Windows Setup boot entry. This process requires the system to be rebooted twice.
7. Reboots the server.

The script requires the following user input:

- Boot device

  You must specify the device path of the disk on which the operating system is to be installed. The script captures this data in the DISK variable. You can specify this value either in the `sstk.conf` file or respond to a prompt during installation.

  For example: `DISK = /dev/cciss/c0d0`

- Boot controller

  The boot controller Windows driver is automatically detected by SSTK. To overwrite the driver used, you must specify the variable `CONTROLLER` in the `sstk.conf` file.

  For example: `CONTROLLER = "retail"`

- Partition sizes

  The ESP and HPSP partitions have a fixed size of 100 MB and 400 MB respectively. The size of the MSR partition depends on the size of the disk:

- On drives less than 16 GB, the MSR is 32 MB
- On drives greater than or equal to 16 GB, the MSR is 128 MB

The `windows-setup.sh` script defines the size of the MSDATA partition as the remainder of free space on the disk after the MSR on disks not greater than 32 GB. You can change the size of this partition as needed. The script uses the `parted` utility to create the disk partitions.

NOTE: The `parted` utility uses decimal instead of binary units to measure disk size. This means that the `parted` command treats 1 KB as 1000 bytes, 1 MB as 1000 KB, and 1 GB as 1000 MB. Other programs calculate this conversion based on multiples of 1024 (2^10). As a result, when you specify a size as an argument to the `parted` command, the actual size is slightly less than you expect.

NOTE: Due to a known issue in mount-to-mount FAT partitions greater than 32 GB, the `parted` utility will create a 32 GB partition on disks with more capacity, and Windows setup will resize the partition to the rest of the disk.

4 Setting up the boot mechanism

Setting up the boot mechanism involves one of the following options: creating a bootable CD or DVD, creating a bootable USB device, or setting up a network boot environment. This chapter shows you how to perform each task at either a Windows workstation or a Linux system. On a Windows workstation, you can create a bootable CD or DVD using any CD or DVD-burning application. This chapter illustrates this task using Nero and Roxio.

Using a bootable CD/DVD

This section shows you how to use a bootable CD or DVD to launch the SSTK. You can use either a Windows workstation or a Linux system to perform the tasks.

From a Windows workstation

The following sections show you how to use a bootable CD/DVD from a Windows system.

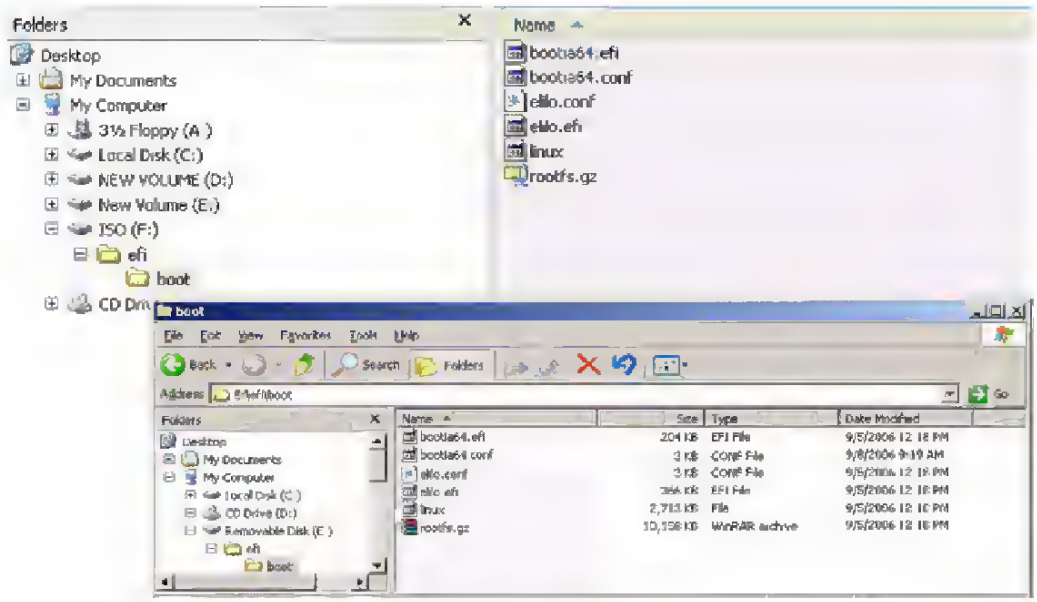
Create a bootable disk image

The SSTK includes the files you need to create a bootable disk image. Creating the disk image involves copying these boot files to a small FAT partition or USB device and using a program such as WinImage (http://www.winimage.com) to create the image (`.ima`) file. The size of the image file depends on the size of the partition. If you create an image from a partition of 50 MB, you will end up with an `.ima` file of 50 MB. Therefore, do not use a partition greater than the capacity of a CD (700 MB). HP suggests a partition size in the range of 20 MB to 512 MB.

To create an image file using WinImage:

1. Copy the contents of the SSTK `boot_files` directory, but not the `boot_files` directory itself, to a FAT partition or USB device, as shown in the following figure.
2. Launch WinImage. From the **Disk** menu select the partition containing the boot files.
3. Select **File->New**, select **Import format from currently selected disk**, and click **OK**.
4. In the **Save As** dialog box, select a **`destination`** for the image file, specify the **`name`** of the file (for example, `ssstk.ima`), and click **Save**.
5. From **Windows Explorer**, drag the entire `efi` directory (Figure 4-1 "EFI Directory") from the partition (F:) and drop it into the right pane of the WinImage main screen.
6. Click **Yes** to finalize the image creation.

Figure 4-1 EFI Directory

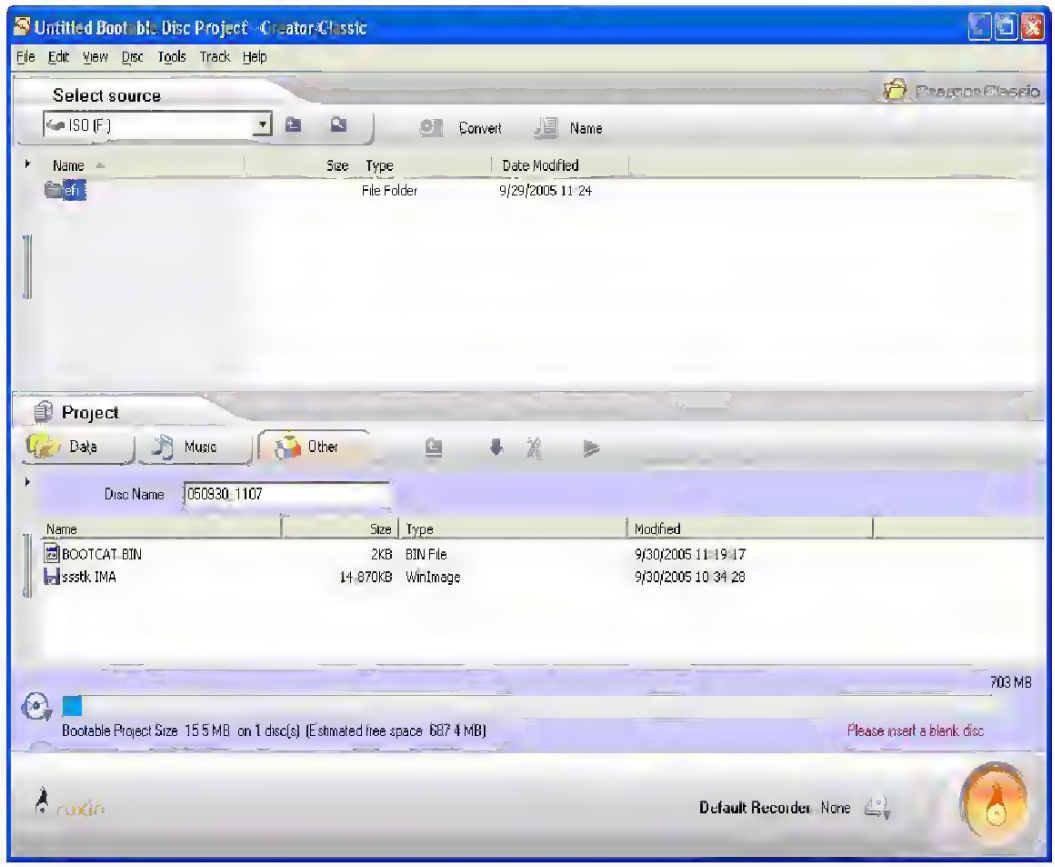


Create the ISO using Roxio

This section shows you how to use the Roxio Easy CD Creator program to create an ISO file containing the bootable image along with the files used by the toolkit.

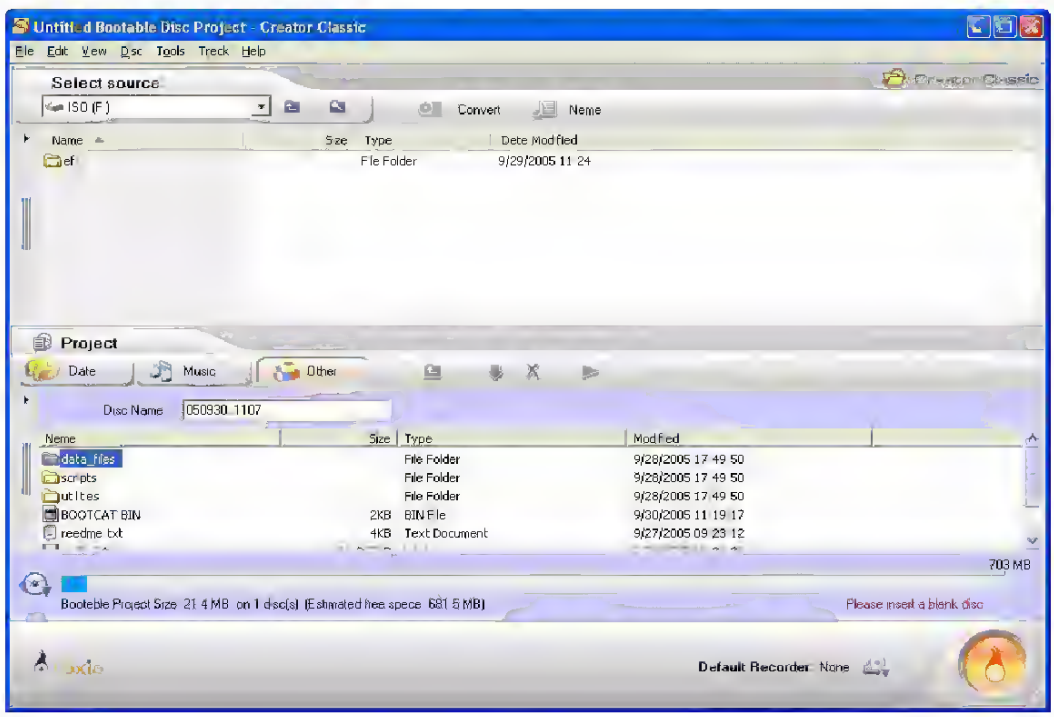
1. Launch Easy CD Creator and select **Disc Creator Classic** (Figure 4-2).
2. When the main screen is loaded, select **File->New Project->Bootable Disc.**
3. From the **Bootable Disc Type** drop down menu, select **No Emulation.**
4. In **Emulation Option,** click **Browse** to select the image file. The Easy CD Creator main screen shows the image file and a BIN file (an auto-generated catalog file).

Figure 4-2 Disc Creator Classic Main Window



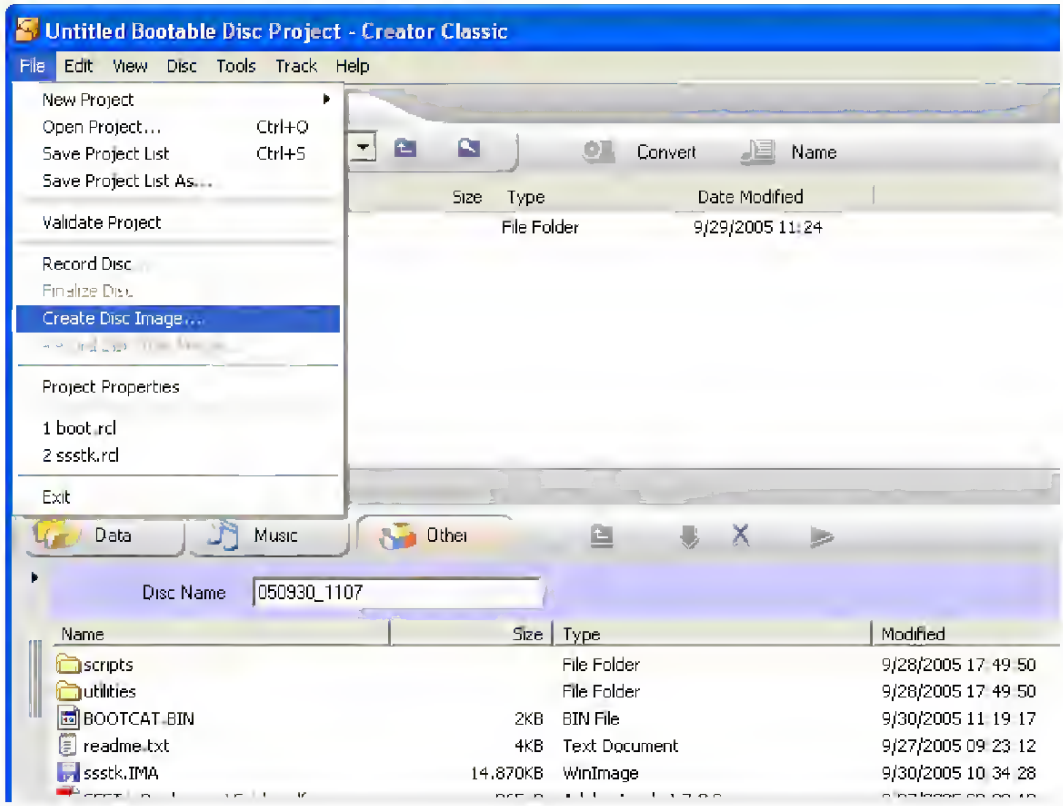
5. Drag and drop the other files you want on the CD to the Project area (Figure 4-3).

Figure 4-3 Roxio Project Area



6. Select **File->Create Disc Image** (Figure 4-4). In the dialog box, enter a file name, select the path, and click **Save** to finalize.

Figure 4-4 Create Disc Image



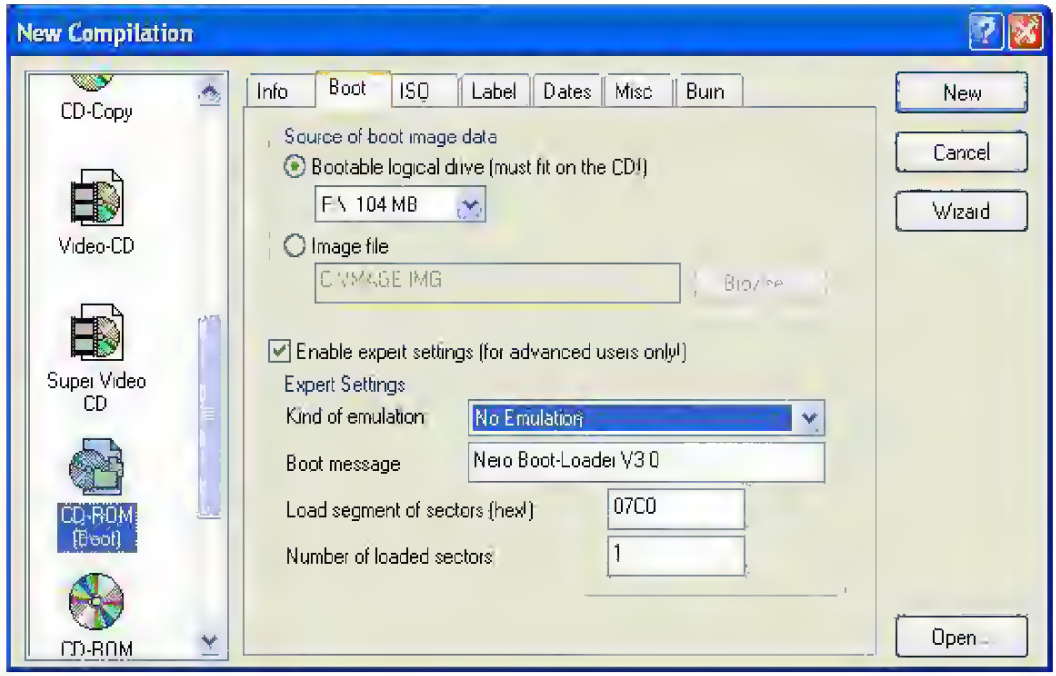
7. Select **File->Burn Disc from Image**.

Create the ISO using Nero

To use Nero to create the ISO file:

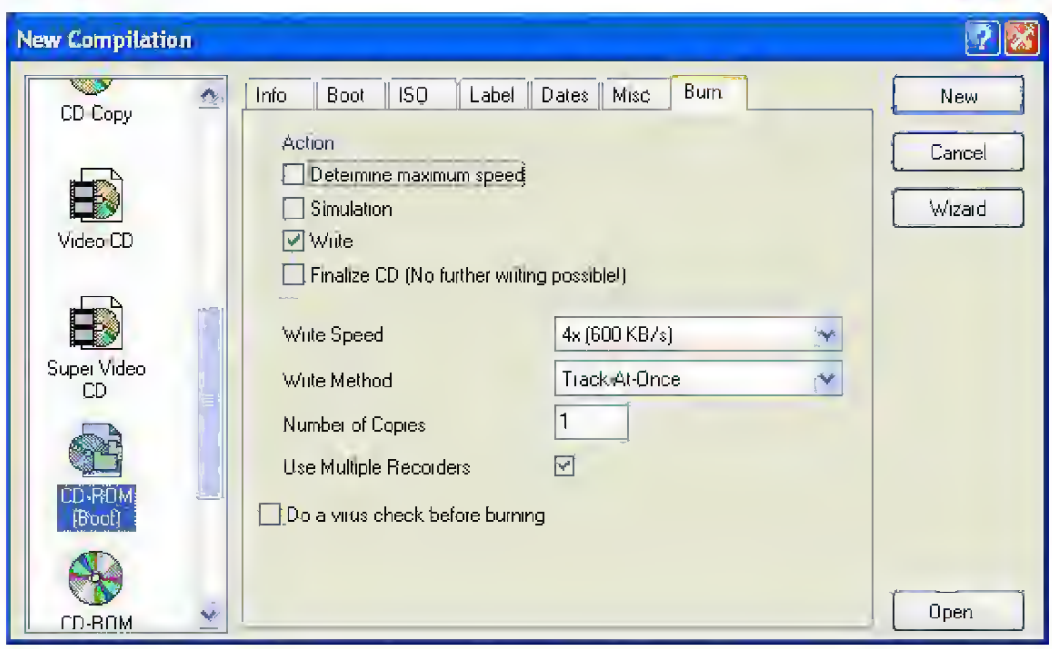
1. Launch Nero, select the **Data** icon, and select **Make a data disc**.
2. Select **File->New**. In the New Compilation screen (Figure 4-5), select the **CD-ROM (Boot)** icon.
3. In the Source of boot image data group, select **Bootable logical drive**. Use the drop down menu to select the drive where the boot files are located. Ensure **Enable expert settings** is checked. From the **Kind of emulation** dropdown menu, select **No Emulation**.

Figure 4-5 New Compilation Window



4. Select the **Burn** tab (Figure 4-6).
5. In the **Action** group, ensure only the **Write** option is checked. From the **Write Method** dropdown menu, select **Track-At-Once**. Check **Use Multiple Recorders**.

Figure 4-6 Burn Tab



6. Click **New**. The **Nero Burning ROM** screen appears.
7. Copy the image files to the left pane of the **Nero Burning ROM** explorer.
8. Click **burn the current compilation** or click **Recorder->Burn compilation**.

From a Linux workstation

The following sections show you how to create a bootable CD/DVD from a Linux system.

Create a bootable disk image

The SSTK includes the files you need to create a bootable disk image. Creating the disk image involves copying these boot files to a small FAT partition and creating the image (`.img`) file.

The bootable disk image must be a FAT file system to be available at the EFI level. Also, if a bootloader file named `bootia64.efi` is located in `\efi\boot` directory, the Integrity server automatically executes that boot loader.

To create a bootable disk image:

1. Create a directory for the bootable image:
   ```
   mkdir ./bootable-image
   ```
2. Use the `dd` command to create a blank file with the size needed to store the boot files:
   ```
   dd if=/dev/zero of=./bootable-image/boot.img bs=1024 count=25000
   ```
3. Format this image file with FAT file system:
   ```
   mkfs.msdos ./bootable-image/boot.img
   ```
4. Create a directory and mount this image file, for example:
   ```
   mkdir ./bootable-image/image
   mount ./bootable-image/boot.img ./bootable-image/image -o loop
   ```
5. Create the bootable directory (`/efi/boot`):
   ```
   mkdir -p ./bootable-image/image/efi/boot
   ```
6. Copy the necessary files from `boot_file/efi/boot` to the `./bootable-image/image/efi/boot` directory:
   ```
   ./bootable-image/image/efi/boot directory:
   bootia64.efi
   bootia64.conf
   elilo.efi
   elilo.conf (bootloader configuration file)
   rootfs.gz (root file system)
   linux (kernel)
   ```
7. If needed, copy additional files to `./bootable-image/image` directory. These files might include toolkit files, configuration files, or third-party tools.
8. Unmount the image file.
   ```
   umount ./bootable-image/image
   ```

The `boot.img` file is now a FAT file system containing the `bootia64.efi` file, which boots the Linux operating system.

Create a CD build directory

1. Create a directory:
   ```
   mkdir ./linuxbootCD
   ```
2. Create a subdirectory to store the boot image:
   ```
   mkdir ./linuxbootCD/boot
   ```
3. Copy the boot image file, `boot.img`, to the `./linuxbootCD/boot` directory:
   ```
   cp /bootable-image/boot.img ./linuxbootCD/boot
   ```

Create the ISO file

The `mkisofs` command is used to create an ISO file. The following table describes the arguments used with this command.

Table 4-1 `mkisofs` command options

Option	Description
`-o linuxbootCD.iso`	Specifies the ISO file
`-b boot/boot.img`	Specifies `boot.img` as the bootable disk image
`-V LinuxBootCD`	Specifies the volume label of the CD.
`./linuxbootCD`	Specifies the target directory as the root of the CD.

1. To create the ISO image, execute the following command at the shell prompt:

```
mkisofs -J -iso-level 3 -r -R -o linuxbootCD.iso \
-b boot/boot.img -c boot/boot.cat \
-V LinuxBootCD -no-emul-boot ./linuxbootCD
```

2. The ISO file can now be written to a CD. The CD shows the following files if viewed from a Windows operating system:

```
boot\boot.imgboot\boot.cat
```

Booting from the CD/DVD

1. Insert the CD in the CD drive Integrity server.
2. Reboot the machine.
3. Enter the EFI Boot Manager and select the Internal Bootable DVD.
4. The server will boot until it reaches the `ELILO boot:` prompt. You have the option of pressing the **TAB**key to view the boot labels of the `elilo.conf` file or wait 30 seconds for the default mount option to start.
5. Select an option created as detailed in "Edit elilo.conf" (page 42).

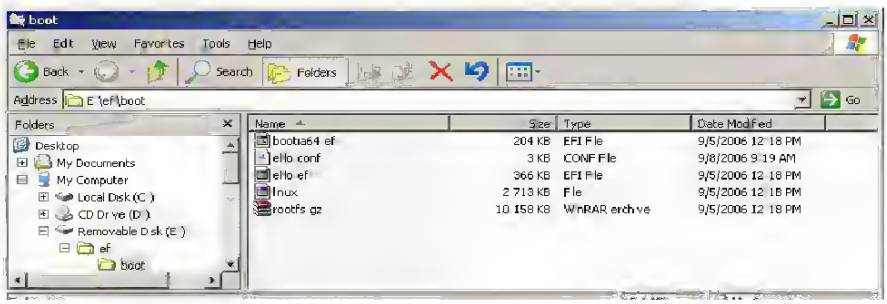
Using a USB flash device

For Integrity servers to boot from a USB flash device, the bootloader and all other files needed to load the Linux image must be copied to the `\efi\boot` directory on the flash device.

Creote o bootoble USB flosh device on o Windows workstotion

The SSTK includes the files you need to create a bootable USB device. Copy the contents of the SSTK `boot_files` directory (Figure 4-7), but not the `boot_files` directory itself, to a the USB device.

Figure 4-7 Copy Boot files



Creote o bootoble USB flosh device on o Linux workstotion

To create a bootable USB flash device:

1. Insert the USB flash device and determine the name of the device.
2. Format the USB flash device with FAT file system:

   ```
   mkdosfs  -I /dev/sda1
   ```

3. Create a directory and mount the USB flash device on this directory:

   ```
   mkdir -p /mntmount /dev/sda1 /mnt
   ```

4. Create the bootable directory for Integrity servers (`/efi/boot`):

   ```
   mkdir -p /mnt/efi/boot
   ```

5. Copy the boot files from the toolkit tar file to the `/mnt/efi/boot` directory:

   ```
   bootia64.efi
   elilo.efi
   elilo.conf (bootloader options)
   rootfs.gz (Linux file system)
   linux (Linux kernel)
   ```

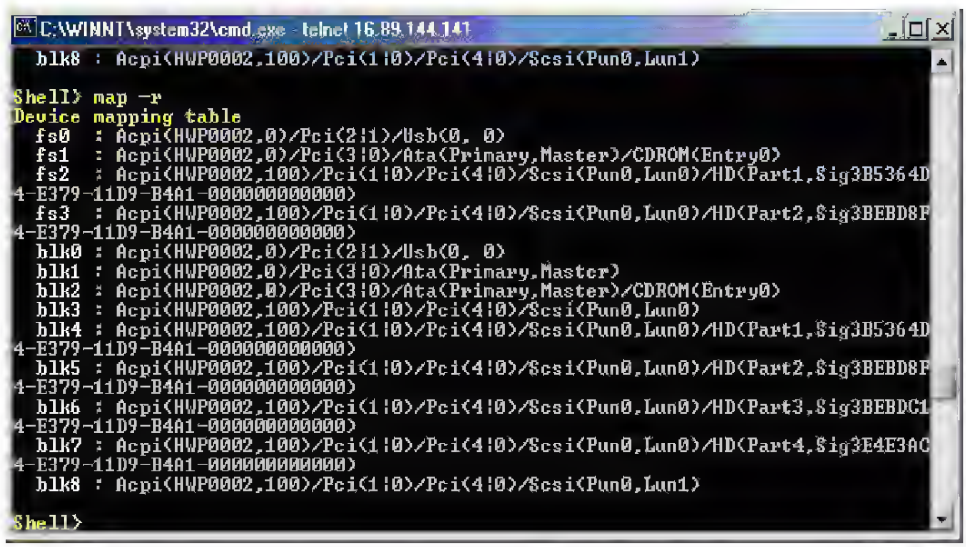
6. Unmount the USB device.

   ```
   umount /mnt
   ```

Boot from the USB flosh device

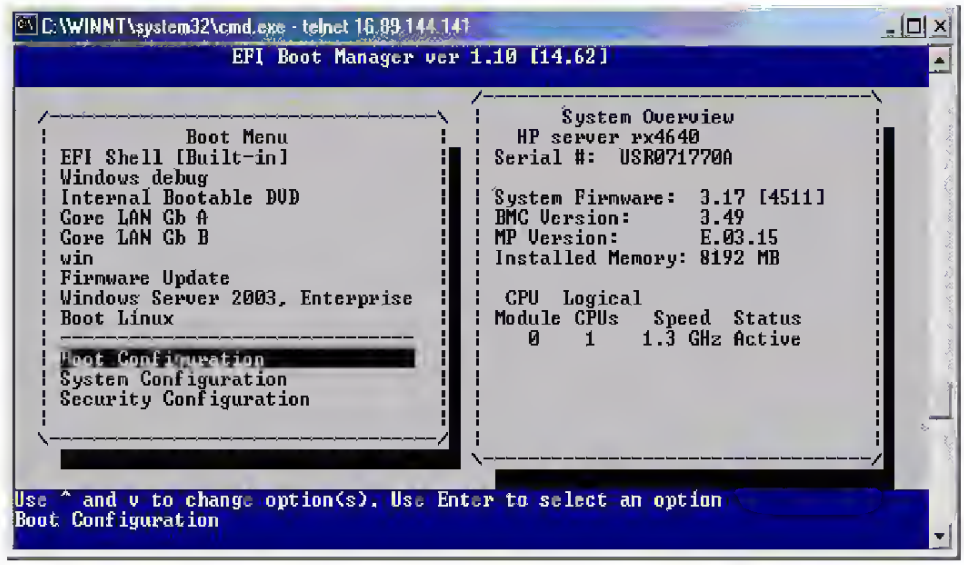
1. Insert the USB flash device into an Integrity server.
2. From the EFI shell, enter `map -r` (Figure 4-8) to refresh the USB device discovery. The USB device should display with a file system.

Figure 4-8 EFI shell Window



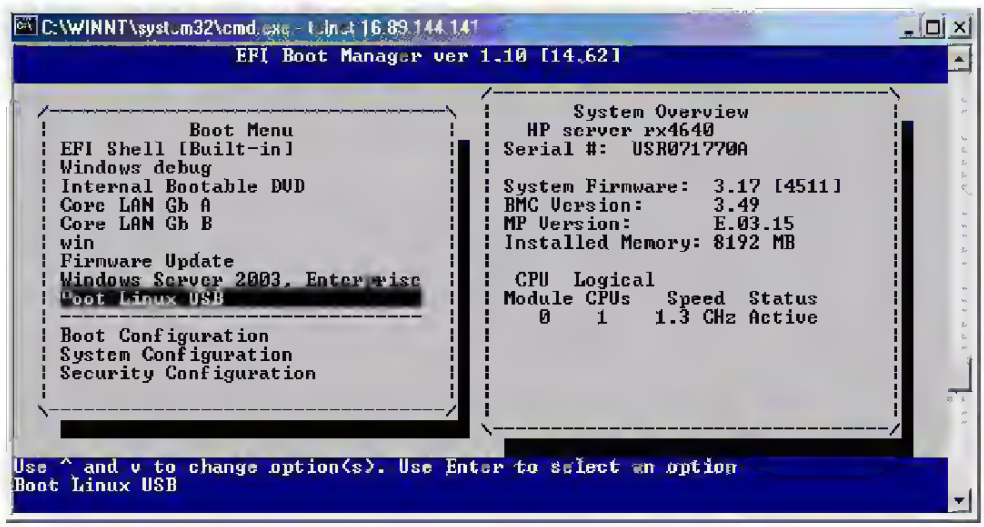
3. Enter the EFI Boot Manager menu, and select **Boot Configuration** (Figure 4-9).
4. Select **Add Boot Option** to add Removable Media Boot for the USB device.

Figure 4-9 Boot Configuration Selection Window



5. In the EFI Boot Manager, select to boot from the Linux USB device (Figure 4-10).

Figure 4-10 Boot from Linux USB Device Window



Using USB flash device as external media

If the USB device is `/dev/sda`, you can mount your MS-DOS formatted USB device using the following command:

```
mount -t msdos /dev/sda /mnt
```

Using network boot

The following sections show you how to set up a network boot environment using a Windows system or a Linux system.

Setting up o Windows network boot server

The DHCP server leases dynamic IP addresses. An extension to the DHCP protocol, PXE, allows a system to boot from an initial operating system image bootstrap program downloaded from a TFTP server. To boot a remote operating system image:

1. The client machine boots with a NIC as the first boot device.
2. The NIC client issues a DHCP DISCOVER.
3. The server replies with a DHCP OFFER. Along with the network configuration information, the server sends information regarding the file to be downloaded by the client:
   - The TFTP server from which the file has to be downloaded.
   - The file name `bootia64.efi`.
4. The NIC downloads the image file from the TFTP server and executes it.
5. The `bootia64.efi` utility obtains its configuration file, `bootia64.conf` , from the same TFTP server, parses it, and executes the configured boot loader (`elilo.efi`).
6. The `elilo.efi` utility automatically obtains its configuration file (usually `elilo.conf`), parses it, downloads the kernel image and `initrd` file from the TFTP server, and starts the boot process.

Setting up a TFTP server

The role of the TFTP server is to provide access to all the files required during the boot process. Copy the files in the `boot_files\efi\boot` directory of the repository to the root directory of the TFTP server or to the path specified in the bootfile name configuration of the DHCP server. The TFTP server must be listening to the default TFTP port (69).

Setting up a DHCP server

The DHCP server must be configured to provide the name of the image to be downloaded by the Integrity server. The following options must be configured in the DHCP server:

- Option 66 — TFTP server name: host name or IP address of the machine running the TFTP server. This must be set even if the TFTP server is running on the same machine as the DHCP server.
- Option 67 — Bootfile name: file name of the boot image. In this application this is `bootia64.efi`. If the file is not located in the root of the TFTP server, the complete path from the root directory must be given. Also, in this case all the remaining files (`bootia64.efi`, `bootia64.conf`, `elilo.efi`, `elilo.conf`, kernel image, and initial RAM disk) must be located in the same directory.

Setting up o Linux network boot Server

This section contains instructions for performing a Linux boot over a network.

NOTE: File locations and process names in this section may vary slightly depending on your Linux distribution.

Setting up a DHCP server

To set up a DHCP server, create and edit a `dhcpd.conf` file in the `/etc` directory of your server. The following is an example assigning a fixed address in a `dhcpd.conf` file:

```
allow booting;
allow bootp;
ddns-update-style ad-hoc;
[Insert the usual DHCP directives, IP ranges, subnet masks, and so on here.]
group {
next-server IP_ADDRESS_OF_TFTP_SERVER;
filename "bootia64.efi";
```

```
host <hostname> {
fixed-address <IP>;
hardware ethernet <MAC>
}
.
.
.
}
```

The `next-server` command tells the DHCP client where to send the TFTP request. The `filename` command tells the DHCP client which file to get. In this example, the file is `bootia64.efi`. When you have finished editing the `dhcpd.conf` file, restart the `dhcpd` process:

```
/etc/init.d/dhcpd restart
```

Setting up a TFTP server

Most Linux installations include a TFTP server and an automated method of launching the server upon receiving a TFTP request. The parent process for detecting a TFTP request and launching the TFTP server is called `xinetd`. However, you might have to enable the TFTP service. The TFTP file is located in the `/etc/xinetd.d/` directory. The following is a sample TFTP file:

```
# default: off
# description: The tftp server serves files using the \
# trivial file transfer protocol. The tftp protocol is \
# often used to boot diskless workstations, download \
# configuration files to network-aware printers and to \
# start the installation process for some operating systems.
service tftp
{
socket_type = dgram
protocol = udp
wait = yes
user = root
server = /usr/sbin/in.tftpd
server_args = -s /tftpboot
disable = yes
per_source = 11
cps = 100 2
}
```

In this example, "disabled" is the default setting, and `/tftpboot` is the root directory for all client access. To enable the TFTP service on your server, edit the disable line to read:

```
disable = no
```

Populating the TFTP directory share

Use the `/tftpboot` directory as the directory from which TFTP clients will get their files. To populate the TFTP directory share:

1. Create a `/tftpboot` directory, if needed.
2. Copy the necessary boot files to the `/tftpboot` directory:

   ```
   bootia64.efi (the boot loader file)
   bootia64.conf (the boot loader configuration file)
   elilo.efi
   elilo.conf (the ELILO configuration file)
   rootfs.gz (the Linux file system)
   linux (the Linux kernel)
   ```

3. The `/tftpboot` directory should now contain the following items:

   ```
   /tftpboot/bootia64.efi
   /tftpboot/bootia64.conf
   /tftpboot/elilo.efi
   /tftpboot/elilo.conf
   ```

```
/tftpboot/rootfs.gz
/tftpboot/linux
```

4. Make sure the `ELILO.conf` file is configured as stated in Chapter 3 (page 41).

Setting up the Integrity server for PXE-enabled boot

Be sure the Integrity server has a PXE-compliant NIC and is physically connected to a configured DHCP server. Set the NIC as the default boot device or select it in the EFI Boot Manager Menu:

1. Boot the server to EFI, and access the EFI Boot Manager menu.
2. Select **Boot Configuration**.
3. Select **Add Boot Entry**.
4. Select **Load File [core LAN GB A]**, where A represents the NIC.
5. Enter a name for the boot entry, and click **Save**.
6. "Core LAN GB A" or the name you entered is listed as an option in the EFI Boot Manager menu. Select this option in the boot menu, and press the **Enter** key.
7. After PXE boots over the network, you have the option of pressing the **Tab** key to view the mounting options of the `ELILO.conf` file.

Setting up a Cellular-based Integrity server for PXE-enabled boot

Be sure that the Cellular-based Integrity server has a PXE-compliant NIC and is physically connected to a configured DHCP server. Set the NIC as the default boot device or select it in the EFI Boot Manager menu:

1. Boot the server to EFI, and access the EFI Boot Manager menu.
2. Select **Boot Option Maintenance Menu->Add a Boot Option**.
3. Select **Load File [Acpi..../Mac(xxxx)]**, where Mac represents the Mac address of a NIC.
4. Select **E-Edit** or **N-New** to create a new entry.
5. Enter a new description name. For example, "LAN1."
6. Select **N-No** Directed LAN.
7. Select **N-No** Boot Option.
8. Select **Y-Yes** to save changes to NVRAM.
9. Select **Exit** until you access the EFI Boot Manager menu.
10. "LAN1" or the name you entered is listed as an option in the EFI Boot Manager menu. Select this option in the boot menu, and press the **Enter** key.
11. Press the **Enter** key.
12. After PXE boots over the network, you have the option of pressing the **Tab** key to view the mounting options of the `ELILO.conf` file.

Booting the Integrity server from a PXE-enabled NIC

1. Enter the EFI Boot Manager and select the appropriate boot core LAN XX option.
2. Use the **Tab** key to see the boot labels defined in the `ELILO.conf` file.
3. Select an option created as detailed in "Edit elilo.conf" (page 42).

Interrupting the SSTK deployment process

If you interrupt the SSTK deployment process on a cell-based server you must consider the ACPI mode. ACPI (Advanced Configuration and Power Interface) defines hardware and software interfaces that enable the operating system to enumerate and configure motherboard devices, and to manage their power.

On a cell-based Integrity server, a special system ACPI flag must be properly set before the operating system is started. When SSTK is running it always sets the ACPI mode to the single-pci-domain or default state and sets to the Windows mode when deploying Windows.

If you are MANUALLY rebooting the server to start the Windows operating system, change the ACPI mode to windows using `hpsetacpi` from SSTK environment (LinuxPE) or the `acpiconfig` utility from EFI environment.

Failure to set the ACPI mode to Windows will cause Windows boot to fail.

NOTE: The SSTK 2.8 utility, `bootia64.efi`, automatically sets the ACPI mode back to Windows after running SSTK if both of the following conditions are met:

- A version of Windows is installed on the system.
- The default boot option points to Windows.

Otherwise, you must set the ACPI flag manually.

---

5 Utilities Reference

This chapter contains reference information and usage examples for the utilities provided in the toolkit. You can get online help for each utility by using the `-h` or `-?` option.

cfggen

Synopsis

`cfggen`

Description

The `cfggen` utility is a RAID Linux configuration utility for LSI controllers, and a counterpart of `cpqacuxe`. In SSTK this utility is used to apply RAID 1 in LSI 1068 controllers.

Options

`cfggen` *`controller # command parameters ...`*

where *`controller #`* is a number between 0 and 255, *`parameters`* is a command-specific value, and *`command`* is a value listed in the following table.

Table 5-1 cfggen Command Values

Value	Description
`AUTO`	Creates an IR volume without specifying drives
`DELETE`	Sets the controller configuration to factory defaults
`DISPLAY`	Displays controller, volume, and physical device information
`LIST`	Lists all available LSI adapters (does not require *`controller #`*)

Examples

Example 5-1 cfggen list command

```
SSTK-v.2.6-4# cfggen list
LSI Logic IR Configuration Utility 2.00.14

          Adapter    Vendor  Device  Segment                          SubSys  SubSys
 Index      Type       ID      ID       ID      Bus  Device  Func  Ven ID  Dev ID
 -----  ----------  ------  ------  --------  ---  ------  ----  ------  ------
   0      SAS1068   1000h    54h    0000h     02h    01h    00h   103ch   170dh
```

Example 5-2 cfggen display command

```
SSTK-v.2.6-4# cfggen 0 display
LSI Logic IR Configuration Utility 2.00.14
Read configuration has been initiated for controller 0
------------------------------------------------------------------------
Controller information
------------------------------------------------------------------------
  Controller type                         : SAS1068
  BIOS version                            : 2.00.14.00
  Firmware version                        : 1.10.01.00
  Channel description                     : 1 Serial Attached SCSI
  Initiator ID                            : 63
  Maximum physical devices                : 62
  Concurrent commands supported           : 511
  Slot                                    : Unknown
```

```
  Bus                                     : 2
  Device                                  : 1
  Function                                : 0
  RAID Support                            : Yes
-------------------------------------------------------------------------
IR Volume information
-------------------------------------------------------------------------
-------------------------------------------------------------------------
Physical device information
-------------------------------------------------------------------------
Initiator at ID #63
Target on ID #1
  Device is a Hard disk
  Enclosure #                             : 1
  Slot #                                  : 1
  Target ID                               : 1
  State                                   : Ready (RDY)
  Size (in MB)/(in sectors)               : 57231/117210240
  Manufacturer                            : ATA
  Model Number                            : FUJITSU MHT2060B
  Firmware Revision                       : 034E
  Serial No                               : NR0LT5625AJP
  Drive Type                              : SATA
Target on ID #2
  Device is a Hard disk
  Enclosure #                             : 1
  Slot #                                  : 2
  Target ID                               : 2
  State                                   : Ready (RDY)
  Size (in MB)/(in sectors)               : 57231/117210240
  Manufacturer                            : ATA
  Model Number                            : FUJITSU MHT2060B
  Firmware Revision                       : 034E
  Serial No                               : NR0LT5625AAG
  Drive Type                              : SATA
-------------------------------------------------------------------------
Enclosure information
-------------------------------------------------------------------------
Enclosure#                                : 1
  Logical ID                              : 50060b00:00c13846
  Numslots                                : 8
  StartSlot                               : 1
  Start TargetID                          : 0
  Start Bus                               : 0
-------------------------------------------------------------------------
```

CLMChecking

Synopsis

```
CLMChecking
```

Description

The `CLMChecking` utility performs memory optimization in cellular systems.

On cellular systems the available system memory can be accessed using two different approaches: interleaved memory, meaning that the average memory latency is constant for all processes in the system; and non-interleaved memory, which provides lower memory latency for processes running in the same cell as the memory.

For each cell, it is possible to determine the amount of memory that should not be interleaved. The non-interleaved memory is known as cell local memory (CLM) or global non-interleaved (GNI) memory. The amount of cell local memory can affect directly the system overall performance. Running Windows, better performance can be obtained with maximum CLM.

Options

Table 5-2 CLMChecking Options

Option	Description
`-p` *`PARTITION`*	Prints the CLM values of the cells in the *`PARTITION`* partition
`-v | --version`	Prints the CLMChecking version.
`-h | --help`	Prints this help

Example

```
SSTK-v.2.7-8# ./CLMChecking
Partition #0:
CLM Cell 0 = 7
CLM Cell 2 = 7

Partition #1:
CLM Cell 1 = 7
CLM Cell 3 = 7
```

conrep

Synopsis

```
conrep ( -l [-f DAT file] [-F] | -s [-x XML FILE] [-f DAT file] | -c |
-d | -h ) [-v]
```

Description

The conrep utility replicates the configuration data stored in the EFI variables. It reads the EFI variables from the NVRAM of the model server and writes the results to a DAT file. It also reads the DAT file and writes this data into the NVRAM of the target server. The conrep utility uses the conrep.xml file to determine what data to retrieve from the server. You can edit this file to control the data collection.

Options

Table 5-3 conrep Options

Option	Description
-s | --read	Reads the EFI environment variables from a server and saves them to the specified file
-l | --write	Loads the EFI environment variables from the specified file and writes them to the server
-m | --reset-mode	Resets ACPI mode to default
-x*filename*	This argument defines the name and location of the XML definition file. The default is conrep.xml
-f*filename*	This argument defines the name and location of the data file. The default is conrep.dat
-c	Clears all boot entries related to the given partition
-d	Restores the EFI boot menu to factory defaults
-v	Verbose execution
-h	Displays help information
-F	Forces the creation of the new EFI variables listed in conrep.dat

Examples

Example 5-3 Sample conrep.dat file

```
<Conrep_data>
<Section name="EFI_general_variables">
<TimeOut>54 00 69 ... 07 00 00 00</TimeOut>
<digest>19 74 58 be 5f 61 f7 da c6 17 79 fd 50 fc 5b 45</digest>
</Section>
</Conrep_data>
```

Example 5-4 conrep.xml file

```
 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
-<conrep>
 -<section name="EFI_general_variables">
    <ev>Lang</ev>
    <ev>Default</ev>
    <ev>UIOption</ev>
    <ev>CPUThreadControl</ev>
    <ev>Timeout</ev>
```

```
    <ev>DBPR####</ev>
  </section>
 -<section name="EFI_Non-Cell_variables">
    <ev>AutobootRetry</ev>
  </section>
 -<section name="EFI_Cell_variables">
    <ev>AutobootRetried</ev>
  </section>
</conrep>
```

Return codes

Table 5-4 conrep Return Codes

Value	Meaning	Cause	Action
0	conrep executed successfully		
1	Permission denied	The user is not a system administrator.	Log in as root to run conrep.
2	Usage error	An error occurred while using conrep for some unexpected reason, such as the wrong parameters.	Run conrep -h to get help using conrep.
3	The conrep.dat file not found	The specified conrep.dat file does not exist	Put the specified conrep.dat file in the correct directory.
4	Error generating hash information to the conrep.dat file	This error occurs only if conrep cannot allocate memory to be used to calculate the hash value for the conrep.dat file.	
5	The conrep.dat file is corrupted	The contents of the conrep.dat file and the hash information do not match	Generation another conrep.dat file.
6	At least one EFI variable described in the conrep.dat file was not found in the system	The machine used to capture the EFI variables and generate the conrep.dat file has some EFI variables that do not exist in the current system.	This is not an error, only a warning. No action is required.
7	Error writing an EFI variable	The user does not have permission to write in the raw_var file in the /sys/firmware/efi/vars/VAR_NAME-GUID directory.	No action is required. The variable will not be written and conrepcontinues to execute
8	The conrep.xml file was not found	There is no conrep.xml file in the current directory	Create a conrep.xml file with the EFI variables that should be read from NVRAM
9	At least one EFI variable in the conrep.xml file was not found in the system	The conrep.xmlfile has some variables that do not exist in the current system. Conrep cannot capture these variables	No action is required. This is not an error, only a warning
10	Error reading an EFI variable	The user does not have permission to read the raw_var file in the /sys/firmware/efi/vars/VAR_NAME-GUIDdirectory	If root does not have permission to read the raw_var file of an EFI variable, this variable must be removed from conrep.xml
11	Error writing to the conrep.dat file	The user does not have permission to write to the conrep.dat file	Change the permissions of the conrep.dat file

cpqacuxe

Synapsis

- Capture: `CPQACUXE -C` *`FILENAME`* `[-internal | -external] -E` *`FILENAME`*
- Input: `CPQACUXE -I` *`FILENAME`* `[-internal | -external] [reset] -E` *`FILENAME`*
- Input with reset: `CPQACUXE -reset` *`FILENAME`* `[-internal | -external] [reset] -E` *`FILENAME`*

Descriptian

The cpqacuxe utility operates on the configuration of storage array controllers. The cpqacuxe utility enables the array configuration on one server to be replicated on other servers with similar array storage resources. The cpqacuxe utility is located in the utilities directory in the toolkit as a tar file. The main toolkit launch script expands the tar file before executing it.

The cpqacuxe utility has three modes of operation:

- In capture mode, it reads the configuration of all internal and external array controllers that are connected to a server and saves this data to a file. You can then use this file to replicate the array configuration on other servers that have similar storage resources.
- In input mode, it applies the array configuration that is specified in a file to a target server. You can create this file using the capture mode and leave it unmodified, edit the capture file, or even create the file from scratch.
- In input mode with reset, it applies the array configuration that is specified in a file to a target server, clearing any existing controller configuration with total data loss. You can use this mode to overwrite the current controller array and logical drive settings.

cpqacuxe capture file

The cpqacuxe utility captures the configuration of all array controllers on the server to a capture file. This capture file can then be used in input mode to replicate the configuration to other array controllers. Also, the capture file can be edited and then reused in input mode. You can then use this file to replicate the array configuration on other servers that have similar storage resources.

Use of the `-internal` or `-external` command line switches effectively filter the controllers captured. For example, if the `-external` switch is used, any controllers internal to the host computer are not captured in the output file.

A typical array configuration capture file is similar to the following:

```
Action = Configure
Method = Custom

; Controller Options
; Controller Compaq Smart Array 5300
Controller = Slot 5
ReadCache = 50
WriteCache = 50
RebuildPriority = Low
ExpandPriority = Low

; Array Options
Array = A
OnlineSpare = No
Drive = 2:0,2:1

; Logical Drive Options
LogicalDrive = 1
RAID = 1
Size = 17359
Sectors = 32
```

```
StripeSize = 256
ArrayAccelerator = Enabled

; HBA SSP Specifications
HBA_WW_ID= 10000000C9272B34
ConnectionName= TestConn
HostMode= Windows
```

Options

Table 5-5 cpqacuxe Options

Option	Description
`-C` *FILENAME*	Specifies the capture file. The capture filename is optional. If no capture file is specified, the default capture file is `acucapt.ini` in the ACU working directory.
`-E` *FILENAME*	Specifies the error file. The error filename is optional. If no error file is specified, the default error file is `error.ini` in the ACU working directory.
`-internal`	Limits the capture to internal controllers.
`-external`	Limits the capture to external controllers.
`-h`	Displays help information.

cpqacuxe input file

The `cpqacuxe` utility uses a text file that contains options and parameters required to configure array controllers. Lines of the script file can be blank or in the form `option = value`. The `cpqacuxe` utility parses the file in a case-insensitive manner. Semicolons are used for comments within script files. If the first character in the line is a semicolon, the line is a comment and the `cpqacuxe` utility ignores everything up to the next line. The valid options in script files generated and read by the ACU utility are divided into the following categories:

- **Control options** — Define the overall behavior of the `cpqacuxe` utility when it processes the scripts and stamps configurations. Control options can occur only once in the script file and must be the first options in the file.
- **Controller options** — Define a controller or set of controllers used in the configuration. This set of options must start with the Controller option. All other options up to the next Controller option will be assumed to belong to the controller that is being defined. These options can be specified in any order, but an option from another category of options cannot occur between any two options in this category.
- **Array options** — Specify and describe a particular array in the configuration. These options are invalid if there is not a preceding controller option section. This set of options must start with the Array option, and all the following options up to the next Array or Controller option are assumed to belong to the array being defined. These options can be specified in any order, but an option from another category of options cannot occur between any two options in this category.
- **Logical drive options** — Specify and describe a particular logical drive in the configuration. These options are invalid if there is no preceding array option. This set of options must start with the LogicalDrive option, and all the following options up to the next LogicalDrive, Array, or Controller option are assumed to belong to the logical drive being defined. These options can be specified in any order, but an option from another category of options cannot occur between any two options from this category. In general, Logical Drives must be listed in the input file in increasing order, unless existing drives are being reconfigured without data loss.
- **HBA options** — Specify and describe an HBA SSP configuration. These options are invalid if there are no preceding logical drive options. This set of options must start with the HBA_WW_ID option, and all the following options up to the next LogicalDrive, Array,

Controller, or HBA_WW_ID option are assumed to belong to the HBA being defined. These options can be specified in any order, but an option from another category of options cannot occur between any two options from this category.

Automatic script file format

The automatic script is used to apply automatic configuration rules to unconfigured or configured controllers. If the `-i-with-reset` or `-reset` command line switch is used, the existing controller configuration is cleared with data loss before the new configuration is applied, effectively making it an unconfigured controller. The `cpqacuxe` utility automatic scripting uses the same configuration suggester that is used in the Express path of the `cpqacuxe` utility GUI. Therefore, the automatic configuration rules are the same as the `cpqacuxe` utility GUI Express path.

The following is an automatic script file syntax:

```
Action = Configure|Reconfigure
Method = Auto

; Controller Options
; There can be multiple controller specifications in the file
Controller = All | First | Slot [N][:N] |
SerialNumber [N]| WWN [N] |
IOCabinet [N],IOBay [N],IOChassis [N],
Slot [N],Cabinet [N],Cell [N]
ClearConfigurationWithDataLoss = Yes|No
LicenseKey = XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
DeleteLicenseKey = XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
RaidArrayId = "XXXXXXXXXXXXXXXXXXXX"
ReadCache = 0|10|20|25|30|40|50|60|70|75|80|90|100
WriteCache = 0|10|20|25|30|40|50|60|70|75|80|90|100
RebuildPriority = Low|Medium|High
ExpandPriority = Low|Medium|High
SurfaceScanDelay= N
OnlineSpare = Yes|No
RAID = 0|1|4|5|6|ADG
```

Table 5-6 Automatic script file options

Option	Value
`Action`	Defines if you are changing existing arrays and logical drives or adding new ones. This option is required. `Configure`: Existing arrays are not modified. If unassigned physical drives exist, new arrays and logical drives are automatically created. `Reconfigure`: Existing arrays might be modified in a non-data-destructive manner. If unassigned physical drives exist, existing arrays might automatically be expanded and new arrays automatically created. New logical drives are created on the expanded and newly created arrays. If the `-i-with-reset` or `-reset` command line switch is used, the existing controller configuration is cleared with data loss as the first step in the configurations process, regardless of the setting of the Action mode.
`Method`	Defines whether you are in Automatic or Custom mode. `Auto` is the default value.

Table 5-6 Automotic script file options *(continued)*

Option	Value
`Controller`	Identifies to which controllers to apply the configuration. This option is required. `Slot [N] [:N]`: The internal controller with slot number *N* is used. External controllers can be identified by appending the Port Number. For example, an MSA1500 connected to a SA6402 might be found at "Slot 4 2," where 4 is the internal Slot number and 2 is the SCSI port. `WWN [N]`: The external controller with World-Wide Name (WWN) *N* is used. `SerialNumber [N]`: The shared storage controller with serial number N is used. `All`: All detected internal and external controllers in the system are used. `First`: The first controller found, based on the controller with the lowest PCI slot number. `IOCabinet [N], IOBay [N], IOChassis [N],Slot [N], Cabinet [N], Cell [N]`: The controller identified by the IPF Slot path information is used. Use of the `-internal"` and `-external` command line switches influences what is regarded as the first controller. For example, if the `-external` switch is used, the hrst controller will be the hrst external controller discovered, regardless of the number of internal controllers in the host system.
`ClearConfigurationWithDataLoss`	Specihes if the configuration should be cleared. This option causes data loss since it deletes all logical drives on the controller. Other scripting commands can exist after this option to recreate a new configuration. This option is optional. `Yes`: The configuration is cleared. All arrays and logical drives on the controller are deleted. `No`: The conhguration is not cleared. This is the default option if the command is not specified in the script file.
`RAIDArrayID`	Specifies the RaidArrayID for controllers that support RaidArrayID. This is a user-defined string to identify controllers. Currently, RaidArrayIDs are only supported by Fibre and shared-storage controllers. `"XXXXXXXXXXXXXXXXXXXX"` Varying length string that can consist of any of the following characters: a-z, A-Z, 0-9, ! @ # * ( ) , - _ + : . / [space] This string has a maximum length that varies depending on the type of controller. For, RA4x00 controllers, the maximum length is 24 characters. For, other controllers, the maximum length is 20 characters. The quotes surrounding the string are optional. Using quotes allows leading space characters to be used in the RaidArrayID. This string cannot end in a space character.
`LicenseKey`	Specifies a license key to install a controller feature. `XXXXX-XXXXX-XXXXX-XXXXX-XXXXX` This is a 25-character key separated by hyphens. The hyphens are optional.
`DeleteLicenseKey`	Specifies a license key to uninstall a controller feature. If the license key is installed, it will be uninstalled. `XXXXX-XXXXX-XXXXX-XXXXX-XXXXX` This is a 25-character key separated by hyphens. The hyphens are optional.
`ReadCache`	Specifies the Read Cache percentage. This option is not required. If this option is not specified, the Read/Write cache ratio will remain unchanged. This value is a numeric value from 0 to 100. It must form a valid read/write cache ratio that is supported on the controller with the WriteCache option.

Table 5-6 Automotic script file options *(continued)*

Option	Value
`WriteCache`	Specifies the Write Cache percentage. This option is not required. If this option is not specified, the Read/Write cache ratio will remain unchanged This value is a numeric value from 0 to 100. It must form a valid read/write cache ratio that is supported on the controller with the ReadCache option
`RebuildPriority`	Specifies the Rebuild Priority. This option is not required. If this option is not specified, the Rebuild Priority will remain unchanged • Low • Medium • High
`ExpandPriority`	Specifies the Expand Priority. This option is not required. If this option is not specified, the Expand Priority will remain unchanged • Low • Medium • High
`SurfaceScanDelay`	Specifies the Surface Scan Delay in seconds. This option is not required. If this option is not specified, the Surface Scan Delay will remain unchanged `N`. This value is a numeric value between 1 and 30 seconds.
`OnlineSpare`	Specifies if Automatic Configuration should attempt to add spares. This option is not required. This option is ignored in Reconfigure mode if there is an existing configuration on the controller. In this case, the configuration suggester makes spare decisions based on the existing configuration. • `Yes`: Automatic Configuration attempts to add spares to each array. This is the default value for the OnlineSpare option if it is not specified. • `No`: Automatic Configuration does not add spares to each array.
`RAID`	Specifies the RAID level that Automatic Configuration uses. It is not required. If the RAID option is not specified or the specified RAID option is not possible, the default RAID is the highest RAID level that is supported with the current state of the controller. This option is ignored in Reconfigure mode if there is an existing configuration on the controller. In this case, the configuration suggester makes RAID decisions based on the existing configuration. • `ADG`: RAID ADG is equivalent to RAID 6 • `6`: RAID 6 • `5`: RAID 5 • `4`: RAID 4 • `1`: RAID 1 • `0`: RAID 0

Custom script file format

The custom script is used to apply custom configurations to unconfigured or configured controllers. You can specify exactly the configuration needed.

The following is a custom script file syntax:

```
The following is a custom script file syntax:
Action = Configure|Reconfigure
Method = Custom

; Controller Options
; There can be multiple controller specifications in the file
Controller = All|Slot [N][:N]|WWN [N]| First |
SerialNumber [N]|
IOCabinet [N],IOBay [N],IOChassis [N],
```

```
Slot [N],Cabinet [N],Cell [N]
ClearConfigurationWithDataLoss = Yes|No
LicenseKey = XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
DeleteLicenseKey = XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
RaidArrayId = "XXXXXXXXXXXXXXXXXXXX"
ReadCache = 0|10|20|25|30|40|50|60|70|75|80|90|100
WriteCache = 0|10|20|25|30|40|50|60|70|75|80|90|100
RebuildPriority = Low|Medium|High
ExpandPriority = Low|Medium|High
SurfaceScanDelay= N
SSPState = Enable|Disable
PreferredPathMode= Auto|Manual

; Array Options
; There can be multiple array specifications in the file
Array = A|B|C|D|E|F|…Z|a|b|c|d|e|f
OnlineSpare = Port:ID,… | Box:Bay,… | Port:Box:Bay | None | N
Drive = Port:ID,… | Box:Bay,… | Port:Box:Bay,… | N | *
DriveType= SCSI | SAS | SATA

; Logical Drive Options
; There can be multiple logical drive specifications in the file
LogicalDrive = 1|2|3|…32
RAID = 0|1|4|5|6|ADG|AUTO
Size = [N]|Max
Sectors = 32|63
StripeSize = 8|16|32|64|128|256
ArrayAccelerator= Enable|Disable
ResourceVolumeOwner= N
LogicalDriveSSPState = Enable|Disable
SSPAdaptersWithAccess = [N],[N]…|None
PreferredPath = 1|2

; HBA Options
; There can be multiple HBA specifications in the file
HBA_WW_ID = WWN
ConnectionName = UserDefinedName
HostMode = Default|Windows|Windows(degrade
|OpenVMS|Tru64|Linux|Solaris|Netware|HP|Windows_SP2_and_
```

Table 5-7 Custom script file options

Option	Value
`Action`	Defines if you are changing existing arrays and logical drives or adding new ones. This option is required. `Configure`: Existing arrays are not modified. Only new arrays and new logical drives can be created in this mode. `Reconfigure`: Existing arrays and logical drives might be modified in a non-data-destructive manner. You can create new arrays, expand existing arrays, create new logical drives, migrate existing logical drives, and extend existing logical drives. If the `-i-with-reset` or `-reset` command line switch is used, the existing controller configuration is cleared with data loss as the first step in the configurations process, regardless of the setting of the Action mode.
`Method`	Defines if you are in Automatic or Custom mode. Specify `Custom` for a custom script.

Table 5-7 Custom script file options *(continued)*

Option	Value
`Controller`	Identifies to which controllers to apply the configuration. This option is required `Slot [N] [:N]`: The internal controller with slot number *N* is used. External controllers can be identified by appending the Port Number. For example, an MSA1500 connected to a SA6402 might be found at "Slot 4:2," where 4 is the internal Slot number and 2 is the SCSI port `WWN [N]`: The external controller with World-Wide Name (WWN) *N* is used. `SerialNumber [N]`: The shared storage controller with serial number N is used `All`: All detected internal and external controllers in the system are used. `First`: The first controller found, based on the controller with the lowest PCI slot number. `IOCabinet [N], IOBay [N], IOChassis [N],Slot [N], Cabinet [N], Cell [N]`: The controller identified by the IPF Slot path information is used. Use of the `-internal"` and `-external` command line switches influences what is regarded as the first controller. For example, if the `-external` switch is used, the first controller will be the first external controller discovered, regardless of the number of internal controllers in the host system.
`ClearConfigurationWithDataLoss`	Specifies if the configuration should be cleared. This option causes data loss since it deletes all logical drives on the controller. Other scripting commands can exist after this option to recreate a new configuration. This option is optional. `Yes`: The configuration is cleared. All arrays and logical drives on the controller are deleted. `No`: The configuration is not cleared. This is the default option if the command is not specified in the script file
`RAIDArrayID`	Specifies the RaidArrayID for controllers that support RaidArrayID. This is a user-defined string to identify controllers. Currently, RaidArrayIDs are only supported by Fibre and shared-storage controllers. `"XXXXXXXXXXXXXXXXXXXX"` Varying length string that can consist of any of the following characters: a-z, A-Z, 0-9, ! @ # * ( ) , - _ + : . / [space] This string has a maximum length that varies depending on the type of controller. For, RA4x00 controllers, the maximum length is 24 characters. For, other controllers, the maximum length is 20 characters. The quotes surrounding the string are optional. Using quotes allows leading space characters to be used in the RaidArrayID. This string cannot end in a space character.
`LicenseKey`	Specifies a license key to install a controller feature. `XXXXX-XXXXX-XXXXX-XXXXX-XXXXX` This is a 25-character key separated by hyphens. The hyphens are optional.
`DeleteLicenseKey`	Specifies a license key to uninstall a controller feature. If the license key is installed, it will be uninstalled. `XXXXX-XXXXX-XXXXX-XXXXX-XXXXX` This is a 25-character key separated by hyphens. The hyphens are optional.
`ReadCache`	Specifies the Read Cache percentage. This option is not required. If this option is not specified, the Read/Write cache ratio will remain unchanged. This value is a numeric value from 0 to 100. It must form a valid read/write cache ratio that is supported on the controller with the WriteCache option.
`WriteCache`	Specifies the Write Cache percentage. This option is not required. If this option is not specified, the Read/Write cache ratio will remain unchanged. This value is a numeric value from 0 to 100. It must form a valid read/write cache ratio that is supported on the controller with the ReadCache option.

Table 5-7 Custom script file options *(continued)*

Option	Value
`RebuildPriority`	Specifies the Rebuild Priority. This option is not required. If this option is not specified, the Rebuild Priority will remain unchanged. • `Low` • `Medium` • `High`
`ExpandPriority`	Specifies the Expand Priority. This option is not required. If this option is not specified, the Expand Priority will remain unchanged. • `Low` • `Medium` • `High`
`SurfaceScanDelay`	Specifies the Surface Scan Delay in seconds. This option is not required. If this option is not specified, the Surface Scan Delay will remain unchanged. `N` This value is a numeric value between 1 and 30 seconds.
`SSPState`	Specifies the SSP State for controllers that support SSP that is enabled/disabled on a controller basis. This option is not required. If this option is not specified, the SSP State remains unchanged. Currently, only Fibre and shared-storage controllers support SSP. This command is not valid for the RA4x00 controller because RA4x00 controllers support SSP that is enabled/disabled on a logical drive basis. `Enable` Enables SSP for the controller. `Disable` Disables SSP for the controller. SSP only remains enabled if the `SSPAdaptersWithAccess` command specifies an adapter for one or more logical drives. If no adapters with access are specified for any logical drives, the SSP State reverts back to disabled.
`PreferredPathMode`	Specifies the preferred path mode when redundant array controllers are in an active-active configuration. This option is only available for certain redundant array controllers. If the option is not specified, it remains unchanged. This option is ignored in active-standby configurations. This option is optional. • `Auto`: This is the default value. The array controller automatically determines which logical drives are assigned to each active array controller for I/O. The array controller dynamically load balances the logical drive paths. • `Manual`: Manually specify which logical drives are assigned to each active array controller for I/O. The array controller does not change the paths. The paths are specified using the `PreferredPath` command in the LogicalDrive section.
`Array`	Specifies the Array ID. `[A-Z, a-f]`: In Configure mode, a new array is created. The array letter specified must be the next available array letter in the existing configuration. In Reconfigure mode, the array letter can identify an existing array, or it can identify the next available array letter in the existing configuration to create a new array.
`Drive`	Specifies the physical drives for the array. A comma separated list specifies the location of physical drives that are to be used in the array. In Configure mode, the physical drives listed are used to create the new array. In Reconfigure mode, all physical drives that are in the array must be listed. Any additional physical drives that are listed are added to the array for an array expansion. • `[X]:[Y], ...`: The Port:ID for controllers that use Port/ID drive numbering schemes or the Box:Bay for controllers that use Box/Bay numbering schemes. • `[X]:[Y]:[Z],...`: The Port:Box:Bay for SAS controllers. • `N`: The number of spares that will be added to the array. The `cpqacuxe` utility select the best available unused physical drives for the spares. • `*`: All available physical drives will be added to the array. Specifying this option consumes all drives and makes adding spare drives impossible.

Table 5-7 Custom script file options *(continued)*

Option	Value
`DriveType`	Restricts the array and operations on the array to the specified type of physical drive. • `SCSI`: Use only SCSI drives • `SAS`: Use only SAS drives. • `SATA`: Use only SATA drives.
`OnlineSpare`	Specifies the spare drives for the array. A comma separated list specifies the location of spare drives that are to be used in the array. • `[X]:[Y], ...`: The Port:ID for controllers that use Port/ID drive numbering schemes or the Box:Bay for controllers that use Box/Bay numbering schemes. • `[X]:[Y]:[Z],...`: The Port:Box:Bay for SAS controllers. • `None`: No spares will be added to the array and any existing spares on the array will be removed. In Configure mode if this option is not specified, `None` is the default value. In Reconfigure mode if this option is not specified, the spare state of the array remains unchanged. • `N`: The number of spares that will be added to the array. The `cpqacuxe` utility selects the best available unused physical drives for the spares.
`LogicalDrive`	Specifies the Logical Drive ID. `[1-32]` In Configure mode, a new logical drive is created. The logical drive number specified must be the next available logical drive number in the existing configuration. In Reconfigure mode, the logical drive number can identify an existing logical drive, or it can identify the next available logical drive in the existing configuration to create a new logical drive. In most cases logical drives must appear in the configuration file in increasing order.
`RAID`	Specifies the RAID level for the current logical drive. In Configure mode, the RAID is the RAID for the new logical drive. In Reconfigure mode, if the logical drive already exists and the RAID is different than the existing RAID on the logical drive. The `cpqacuxe` utility then attempts to migrate the logical drive to the specified RAID. The RAID option is not required. If the RAID is not specified for a new logical drive, the RAID will be the highest RAID level that is supported on the current Array. If the RAID is not specified for an existing logical drive in Reconfigure mode, then the current RAID setting remains unchanged. • `AUTO`: The best available RAID level is selected based on the number of disks in the array. This is normally be the highest RAID level supported. • `ADG`: RAID ADG is equivalent to RAID 6. • `6`: RAID 6 • `5`: RAID 5 • `4`: RAID 4 • `1`: RAID 1 • `0`: RAID 0
`Size`	Specifies the size of the logical drive. If the logical drive is new, the size is for the new logical drive. In Reconfigure mode, if the logical drive already exists and the size is different than the existing size of the logical drive, the `cpqacuxe` utility attempts to extend the logical drive to the new size. This option is not required. If the size is not specified for a new logical drive, the default is `MAX`. If the size is not specified for an existing logical drive, the current logical drive size remains unchanged. • `[N]`: The size in MB • `MAX`: Uses the maximum size possible for the logical drive.

Table 5-7 Custom script file options *(continued)*

Option	Value
`Sectors`	Specifies the Max Boot setting (Sectors Per Track) for the logical drive. The Sectors option is not required. If the logical drive is new, the default is 32. If the logical drive already exists, then the default is the current Sectors setting on the logical drive. • `32`: Disable Max Boot (32 sectors per track) • `63`: Enable Max Boot (63 sectors per track).
`StripeSize`	Specifies the Stripe Size for the logical drive in KB. In Configure mode, the Stripe Size is the Stripe Size for the new logical drive. In Reconfigure mode, if the logical drive already exists and the Stripe Size is different than the existing Stripe Size on the logical drive, the `cpqacuxe` utility attempts to migrate the logical drive to the specified Stripe Size. The Stripe Size option is not required. If the Stripe Size is not specified for a new logical drive, the Stripe Size is the default Stripe Size for the RAID level of the logical drive. If the Stripe Size is not specified for an existing logical drive in Reconfigure mode and the user is not migrating the RAID, then the current Stripe Size setting remains unchanged. The following Stripe Size values are available. • `8`: 8 KB • `16`: 16 KB • `32`: 32 KB • `64`: 64 KB • `256`: 256 KB • `512`: 512 KB The following Stripe Size values are available for RAIDS: • RAID 0: `8, 16, 32, 64, 128, 256` (default is 128) • RAID 1: `8, 16, 32, 64, 128, 256` (default is 128) • RAID 4: `8, 16, 32, 64` (default is 16) • RAID 5: `8, 16, 32, 64` (default is 16) • RAID ADG: `8, 16, 32, 64 (default is 16)`
`ArrayAccelerator`	Specifies the Array Accelerator setting for the logical drive. • `Enable`: Enables the Array Accelerator for the logical drive. This is the default setting for new logical drives. • `Disable`: Disables the Array Accelerator for the logical drive.
`ResourceVolumeOwner`	Specifies that this logical drive should be a snapshot resource volume and specifies the owner logical drive ID. If this command is not specified, the logical drive will be a normal data volume. `N`: This value is the logical drive ID of an existing logical drive which owns this snapshot resource volume.
`LogicalDriveSSPState`	Specifies the SSP State for a logical drive. This command is only valid for controllers that support SSP that is enabled/disabled on a logical drive basis. This option is not required. If this option is not specified, the SSP State remains unchanged for existing logical drives and is disabled for new logical drives. Currently, only the RA4x00 controller supports SSP that is enabled/disabled on a logical drive basis. For other controllers that support SSP, see the SSPState command. • `Enable`: Enables SSP for the logical drive. • `Disable`: Disables SSP for the logical drive.
`SSPAdaptersWithAccess`	Specifies the SSP adapters that have access to a logical drive. This command is only valid for controllers that support SSP. This option is processed only if the SSPState or LogicalDriveSSPState is set to Enable. Otherwise, this command is ignored. • `[N],[N]...`: A comma separated list of SSP adapter IDs that are granted access to the logical drive. • `None`: No SSP adapters are granted access to the logical drive.

Table 5-7 Custom script file options *(continued)*

Option	Value
`PreferredPath`	Specifies the preferred path for the logical drive when redundant array controllers are in an active-active redundancy mode and the preferred path mode is set to Manual. This option is only available for certain redundant array controllers. If the option is not specified, the preferred path remains unchanged for the logical drive. This option is ignored in active-standby configurations and in active-active configurations when the preferred path mode is Auto. 1 The controller in chassis-slot 1 is the preferred controller I/O path for the logical drive. This is the default value. 2 The controller in chassis-slot 2 is the preferred controller I/O path for the logical drive.
`HBA_WW_ID`	Specifies to which HBA to apply the configuration changes. `WWN [N]`. The external controller with World-Wide Name (WWN) N is used.
`ConnectionName`	A user defined string used as the connection name for the specified HBA. "XXXXXXXXXXXXXXXXXXXX" Varying length string that can consist of any of the following characters: a-z, A-Z, 0-9, ! @ # * ( ) , - _ + : / [space]. This string has a maximum length of 16 characters and can contain embedded spaces but cannot end in a space character.
`HostMode`	Specifies the Host Mode for the selected HBA. Setting the Host Mode optimizes the storage array for the selected operating system. The available host modes for an HBA are device specific, and not all modes are available on all devices. Not all HBAs support a Host Mode. The following operating system options might be available. • `Default` • `Windows` • `Windows (degrade)` • `OpenVMS` • `TRU64` • `Linux` • `Solaris` • `Netware` • `HP` • `Windows_SP2`

Error reporting

When using the `cpqacuxe` utility, any errors that are encountered are logged to the `error.ini` file or the specified error file. Error codes are also returned to the calling process, allowing scripting clients to key off the return code instead of having to parse the error file.

The error file displays the error code, error message, and if possible, lists the controller, array, or logical drive associated with the error.

The format of the error file is as follows:

```
Line1:ACU Configuration Replication Error
Line2:ERROR: (<error code>) <error description>
Line3:Controller: Slot # | SerialNumber # | WWN #
Line4:Array: #
Line5: Logical Drive: #
```

Table 5-8 cpqacuxe Error Codes

Code	Error	Reason
0	Success	Success
1	General error	Problem spawning ACU process (user is not authenticated to use ACU or ACU is already running)
2866	Failure opening capture file '%s'	Error opening capture file
2867	Failure opening input file '%s'	Error opening input file
2868	Failure opening error file '%s'	Error opening error file
2869	'%s' command expected	The specified command is missing or in the incorrect place in the file.
2870	'%s' is not a supported command	The specified command is not a valid scripting command
2871	'%s' is not a Controller command	The specified command does not belong in the Controller section of the script file
2872	'%s' is not an Array command	The specified command does not belong in the Array section of the script file
2873	'%s' is not a Logical Drive command	The specified command does not belong in the LogicalDrive section of the script file
2874	'%s' is not an HBA command	A command found in the HBA section is not a valid command for this section.
2875	More than one '%s' command cannot exist in the same section	The specified command is used more than once in the same section. It should only be used once per section.
2821	No controllers detected	No controllers were detected. This error applies to input mode only
1102	Slot information is not available	Input mode cannot be run on internal controllers without slot information online. For Windows, this means the System Management Driver has to be loaded.
2819	Controller '%s' is invalid	The controller specified is not specified correctly.
2820	Could not detect Controller '%s'	The controller specified was not detected.
2818	Invalid Method	Invalid Method value
2819	Invalid Controller	Invalid Controller value.
2822	Invalid Read Cache/Write Cache Ratio	The specified Read Cache/Write Cache Ratio is not supported either by the controller or the current controller configuration.
2823	Invalid Rebuild Priority	The specified Rebuild Priority is not supported.
2824	Invalid Expand Priority	The specified Expand Priority is not supported. Also, this error occurs if the controller does not allow expansion and therefore does not support Expand Priority
2857	Invalid Surface Scan Delay	Invalid Surface Scan Delay
2849	Invalid ClearConfigurationWithDataLoss parameter	The specified ClearConfigurationWithDataLoss parameter is not correct
2825	Invalid Array	The Array ID is invalid.

Table 5-8 cpqacuxe Error Codes *(continued)*

Code	Error	Reason
2826	`Array not specified`	The array command is missing from the script file. Some commands were found that require an array to work on.
2827	`New Array ID does not match the next available Array ID`	The Array ID in the script file does not match the Array ID of the newly created array. For example, if you have Array A already and the script file tries to create Array C (without Array B), the script generates this error.
2828	`New Array ID already exists`	When running in Configure mode, this error happens when the Array ID in the script file already exists in the configuration. Configure mode can only create new arrays.
2829	`Cannot create Array`	The controller does not allow a new array to be created. This could be because there are no unassigned physical drives on the controller, or the maximum number of arrays or logical drives has already been reached.
2830	`Cannot expand Array`	The array could not be expanded. This could be because the controller does not support expansion, or the current configuration does not allow expansion to occur on the array.
2831	`Cannot change Array Spare`	The spare state of the array could not be changed. This can happen when you are trying to add a spare or drop a spare. The current configuration does not allow a spare state change for the array.
2832	`Invalid Physical Drive`	The physical drive listed for the array is not a valid physical drive, or it is a physical drive that is not capable of being placed in the array.
2833	`Invalid Spare`	The spare drive listed for the array is not a valid spare drive, or it is a drive that is not capable of being placed on the array as a spare.
1053	`Cannot remove Physical Drives from existing array`	One or more drives are not listed in an existing array. This is illegal because removing physical drives from a configured array causes data loss.
1052	`Array requires an odd number of drives`	This error message can occur if the user is attempting to add an odd number of drives to an existing array that has RAID 1 logical drives. If the controller has RAID 1 logical drives and supports RAID 1 -> RAID 5/RAID ADG migration, this error will not occur. However, if the controller does not support migration, this error occurs.
2834	`Invalid Logical Drive`	The Logical Drive ID is not valid.
2835	`Logical Drive not specified`	The Logical Drive command is missing from the script file. Some commands were found that require a Logical Drive to work on.
2836	`New Logical Drive ID does not match the next available Logical Drive ID`	The Logical Drive ID in the script file does not match the Logical Drive ID of the newly created logical drive. For example, if you already have Logical Drive 1 and the script file attempts to create Logical Drive 3 (without Logical Drive 2), the script generates this error. This error can occur when using an input file with logical drive numbers that are not sequential. In this case, the logical drive numbers need to be changed so that they are sequential in the input file.
2837	`New Logical Drive ID already exists`	When running in Configure mode, this error occurs when the Logical Drive ID in the script file already exists in the configuration. Configure mode can only create new logical drives.
2838	`Cannot create Logical Drive`	The controller does not allow a new logical drive to be created. This could be because there is no free space on the array, or the maximum number of logical drives has already been reached.

Table 5-8 cpqacuxe Error Codes *(continued)*

Code	Error	Reason
2839	`Cannot migrate Logical Drive Raid`	The controller does not support migration, or the current configuration of the controller does not allow migration to occur.
2840	`Cannot migrate Logical Drive Stripe Size`	The controller does not support migration, or the current configuration of the controller does not allow migration to occur
2841	`Cannot extend Logical Drive`	The controller does not support extension, or the current configuration of the controller does not allow extension to occur. For example, there may be no free space on an array, so extension is not supported for the logical drive.
2842	`Invalid Raid`	The specified RAID is invalid or is not possible with the current configuration
2843	`Invalid Size`	The specified size is invalid or is not possible with the current configuration
2844	`Invalid Stripe Size`	The specified stripe size is invalid, is not supported with the current RAID level, or is not supported with the current configuration
2846	`Cannot change Logical Drive Sectors`	You cannot change the max boot on a configured logical drive This leads to data loss.
2845	`Invalid Sectors`	The specified Max Boot setting is invalid, or is not supported with the current configuration
2848	`Cannot change Logical Drive Array Accelerator Setting`	The current controller configuration does not support changing the logical drive Array Accelerator state.
2847	`Invalid Array Accelerator Setting`	The specified Array Accelerator setting is invalid or is not supported with the current configuration.
290	`Error communicating with Controller`	There was a problem communicating with one or more controllers
278-282, 2564	`Controller is locked by another machine or user`	One or more controllers is locked by another machine or user
268	`Error saving Controller`	There was a problem saving one or more controller configurations.
1065-1066	`Too many coinciding expansion, migration, or extension operations`	This error is the "invalid increase/decrease" in logical drive offset error. This can happen when too many unsaved expansions, extensions, or migrations are tried at the same time.
1110	`Controller does not support license keys`	The controller does not support license key entry or deletion
1093	`Controller requires physical drives to set license keys`	The controller requires physical drives for license keys.
1114	`Controller requires non-failed physical drives to set license keys`	The controller requires non-failed physical drives for license keys.
1111	`Invalid license key`	The entered license key is not a valid license key.
1112	`Controller has maximum number of license keys`	The controller already has the maximum number of license keys installed
1110	`License key is not a controller feature license key`	The entered license key is not a controller feature license key
2850	`Controller does not support Raid Array ID`	The controller does not support Raid Array ID.

Table 5-8 cpqacuxe Error Codes *(continued)*

Code	Error	Reason
2851	`Invalid Raid Array ID`	The entered Raid Array ID is invalid. The Raid Array ID can only consist of the following characters: a-z, A-Z, 0-9, ! @ # * ( ) , - _ + . / [space] It cannot end with a space character or exceed the maximum number of characters allowed by the controller.
1091	`Controller does not support SSP`	The controller does not support SSP.
2855	`Controller does not support Logical Drive SSP states. Use 'SSPState' Controller command to set the Controller SSP state`	The Controller does not support Logical Drive SSP states. Use `SSPState` controller command to set the Controller SSP state
2856	`Controller does not support Controller SSP state. Use 'LogicalDriveSSPState' Logical Drive command to set SSP states for each Logical Drive`	Controller does not support Controller SSP state. Use `LogicalDriveSSPState` logical drive command to set SSP states for each Logical Drive
2853	`Cannot change SSP settings`	The SSP settings cannot be changed for the controller or logical drive
2854	`Invalid SSP Adapter ID`	The adapter ID is not a valid adapter ID that was detected by the controller.
2852	`Invalid SSP State`	The requested SSP State is not a valid SSP state.
2858	`Controller does not support snapshot resource volume feature`	The controller does not support the snapshot resource volume feature
2859	`Invalid Resource Volume Owner`	The specified Resource Volume Owner is not valid. The Resource Volume Owner should be a valid existing logical drive ID.
2860	`Cannot change Resource Volume Owner`	The Resource Volume owner cannot be changed if it is a configured logical drive.
2861	`Controller does not support redundancy settings`	The controller is not redundant or does not support redundancy settings
2864	`Invalid Preferred Path Mode`	The preferred path mode is not a valid preferred path mode or the controller is not available
2865	`Invalid Preferred Path`	The preferred path specified is not a valid chassis-slot for an available active controller or the controller is not available
2876	`Invalid Physical Drive Count`	Invalid Physical Drive count. A possible reason is that more disks were requested than available of the current DriveType
2877	`No Spares Available`	No suitable Spares available.
2878	`Spare Request for RAID0 is Invalid`	Spares are not supported in RAID0-only Arrays
2879	`Reset and Reconfigure combined Error`	Mode is Reconfigure and a controller reset with data loss was specified
2880	`Invalid Drive Type Specified`	Invalid Disk Type specified
516	`Internal Error`	An internal ACU error has occurred. This error should never occur. If it does, there is a problem with ACU, and some error is not being identified properly

efibootmgr

Synopsis

```
efibootmgr [command]... [option]...
```

Description

The `efibootmgr` utility is used from a script file to manager the EFI Boot variables. The `efibootmgr` utility is used for creating or destroying the EFI BOOT variables and to update the BOOT NEXT, BOOT ORDER, and TIMEOUT variables. The BOOT ORDER variable defines the boot entries sequence to be shown in menu. The TIMEOUT sets the time-out to select automatically the boot variables, according to BOOT ORDER. The BOOT NEXT sets the next boot entry to be selected automatically disregarding the BOOT ORDER and TIMEOUT.

Options

Table 5-9 efibootmgr Options

Options	Description
`-a` | `--active`	Set bootnum active.
`-A` | `--inactive`	Set bootnum inactive.
`-B` | `--delete-bootnum`	Delete bootnum (hex).
`-c` | `--create`	Create new variable bootnum and add to bootorder
`-h` | `--help`	Display the help and exit.
`-i` | `--iface` *name*	Create a netboot entry for the named interface.
`-n` | `--bootnext` *XXXX*	Set BootNext to *XXXX* (hex)
`-N` | `--delete-bootnext`	Delete BootNext
`-o` | `--bootorder` *XXXX,YYYY,ZZZZ,...*	Explicitly set BootOrder. The values must be in hex.
`-O` | `--delete-bootorder`	Delete BootOrder
`-t` | `--timeout` *seconds*	Set boot manager timeout waiting for user input.
`-T` | `--delete-timeout`	Delete Timeout
`-V` | `--version`	Return version and exit.

Table 5-10 Commands

Commonds	Description
`-@` | `--@ file`	Defines the file to include more options
`-b` | `--bootnum` *XXXX*	Selects BootXXXX (hex) variable to be deleted.
`-d` | `--disk` *disk*	Device file that the boot loader stores (defaults to `/dev/sda`).
`-e` | `--edd [1|3|-1]`	Forces EDD 1.0 or 3.0 creation variables or guess.
`-E` | `--device` *num*	EDD 1.0 device number (defaults to 0x80)
`-g` | `--gpt`	Forces disk with invalid PMBR to be treated as GPT
`-H` | `--acpi_hid` *XXXX*	Sets the ACPI HID (used with `-i`)
`-l` | `--loader` *name*	Boot loader file name. The full path must be informed if necessary (defaults to `\elilo.efi`).

Table 5-10 Commands *(continued)*

Commands	Description
`-L | --label label`	Boot manager display label (defaults to "Linux").
`-p | --part part`	Partition number that containing boot loader (defaults to 1)
`-q | --quiet`	Be quiet.
`--test filename`	Does not write to NVRAM, writes to `filename`.
`-u | --unicode | --UCS-2`	Passes extra args as UCS-2 (default is ASCII).
`-U | --acpi_uid XXXX`	Sets the ACPI UID (used with `-i`).
`-v | --verbose`	Prints additional information
`-w | --write-signature`	Writes unique signature to MBR if needed

Table 5-11 efibootmgr Command Line Input

Command line input	Result
`efibootmgr`	Shows the current configuration
`efibootmgr -c`	Creates a default boot entry with the following values: NAME `Linux` DEVICE `/dev/sda` PARTITION 1 BOOT LOADER `/elilo.efi` LOAD OPTION `none`
`efibootmgr -c -L "New Boot" -l /ia64/setupldr.efi -d /dev/cciss/c0d0 -p 4 WINDOWS`	Creates a boot entry with following values: NAME `New Boot` DEVICE `/dev/cciss/c0d0` PARTITION 4 BOOT LOADER `/ia64/setupldr.efi` LOAD OPTION `WINDOWS`
`efibootmgr -c -L "Boot" -l /setupldr.efi --test bootentry.bin`	Generates a boot entry but does not save it in NVRAM. The boot entry values are: NAME `Boot` DEVICE `/dev/sda` PARTITION 1 BOOT LOADER `/setupldr.efi` LOAD OPTION `none`
`efibootmgr -c -l /setupldr.efi -@ parameter.txt`	Creates a boot entry with the following values: NAME `Linux` DEVICE `/dev/sda` PARTITION 1 BOOT LOADER `/setupldr.efi` LOAD OPTION data into the `parameter.txt` file
`efibootmgr -B -b 0002`	Deletes the second boot entry (BOOT0002)
`efibootmgr -n 0005`	Sets the fifth boot entry (BOOT0005) to BOOT NEXT
`efibootmgr -o 0002,0005,0001`	Defines the new BOOT ORDER value
`efibootmgr -t 5`	Sets time-out to 5 seconds
`efibootmgr -O -N -T`	Deletes the BOOT NEXT, BOOT ORDER and TIMEOUT values

efivar

Synopsis

```
efivar
```

Description

The `efivar` utility is used to set EFI variables from SSTK LinuxPE.

Options

Option	Description
`-g | --guid`	The EFI variable GUID (default is `8be4df61-93ca-11d2-aa0d-00e098032b8c`)
`-n | --name`	The EFI variable name
`-v | --value`	The EFI variable value
`-d | --delete`	Deletes EFI variable
`-U | --unicode`	Converts to Unicode when saving

Example

```
SSTK-v.2.7# ./efivar -n test1 -v "abc def"
name: test1

SSTK-v.2.7# ls /sys/firmware/efi/vars
Boot0000-8be4df61-93ca-11d2-aa0d-00e098032b8c
Boot0001-8be4df61-93ca-11d2-aa0d-00e098032b8c
...
preferred-bsp-7f0b2634-4f16-413d-bd17-80f6bba37a9b
test1-8be4df61-93ca-11d2-aa0d-00e098032b8c

cat /sys/firmware/efi/vars test1-8be4df61-93ca-11d2-aa0d-00e098032b8c/data
abc def
SSTK-v.2.7#
```

gen-setupldr-options

Synopsis

```
gen-setupldr-options -d device [-p number] -w path [-o file]
[-h]
```

Description

The `gen-setupldr-options` utility generates options for the setup loader, which launches the Windows installation process. Use `gen-setupldr-options` along with the `efibootmgr` utility.

Options

Table 5-12 gen-setupldr-options Options

Options	Description
`-d device`	Sets the device file.
`-w path`	Sets the working path.
`-p number`	Sets the partition number.
`-o file`	Sets the output file name. If not defined the output is standard output (`stdout`).
`-h`	Displays the help and exit.

Table 5-13 gen-setupldr-options Commond line input

Commond line input	Result
`gen-setupldr-options -d /dev/sda -w /windows`	Writes to `stdout` the `setupldr.efi` options to launch the Windows installation process in `/windows`directory localized in first partition of the `/dev/sda` device. Return error code 0 (zero).
`gen-setupldr-options -d /dev/cciss/c0d0-p 4 -w $win_nt$.~ls\ia64 -ooptions.bin`	Saves the `options.bin` file the `setupldr.efi` options generated to launch the Windows installation process in the `/$win_nt$.~ls/ia64` directory localized in fourth partition of the `/dev/cciss/c0d0` device. Return error code 0 (zero).
`gen-setupldr-options -d /dev/cciss/c0d0 -p 4 -w $win_nt$.~ls\ia64 -o options.bin`	This command returns the following error codes: 0 - file option.bin successfully generated 1 - if /dev/cciss/c0d0 or win_nt.~ls/ia64 had not been informed 2 -if /dev/cciss/c0d0 does not exist 2 - if /dev/cciss/c0d0 is not a device file 2 - if /dev/cciss/c0d0 have not GPT partition table 2 - if /dev/cciss/c0d0 have not four or more partitions 2 - if /dev/cciss/c0d0 partition 4 is empty 3 - any problem to create options.bin file

getuuid

Synopsis

```
getuuid
```

Description

The `getuuid` utility retrieves the UUID (Universally Unique ID) used to identify a storage device or a single partition.

Options

Option	Description
disk	The disk to retrieve information from
`-p`	Partition number
`-t`	Shows GUID of partition type (EFI, hpsp, and so on)

hpia64info

Synopsis

```
hpia64info -v
```

Description

The `hpia64info` utility displays information about management processor and system firmware.

Options

The `hpia64info` utility has one option, `-v`. This option displays the version of the `hpia64info` utility.

Output

The `hpia64info` utility displays the following management processor information:

- Version
- LAN status
- Host name
- IP address
- Netmask
- Gateway
- MAC address

The`hpia64info` utility displays the following system firmware information:

- Asset tag
- BMC (Baseboard Management Controller) version
- SFW (System Firmware) version
- EFI (Extensible Firmware Interface) version

Example 5-5 Sample hpia64info utility output

```
Format: 1
MP Version: F.01.55
MP Lan Status: Enabled
MP Host Name: superbird
MP IP Address: 16.89.144.239
MP Netmask: 255.255.252.0
MP Gateway: 16.89.144.1
MP MAC Address: 00-14-C2-9C-07-6C
Asset Tag: hello world
BMC Version: 05.07
SFW Version: 01.31
EFI Version: 05.37
```

hplpcfg

Synopsis

```
hplpcfg
```

Description

The `hplpcfg` utility enables or disables the EFI or BIOS boot setting on Emulex 2 GB or 4 GB Fibre Channel Host Bus Adapters (FC HBA).

Options

Table 5-14 hplpcfg Options

Option	Description
`/se filename`	Saves the HBA configuration to *`filename`*
`/le filename`	Loads the HBA configuration from *`filename`*

Format

The configuration file for the EFI boot setting has the following format.

Formot	Description
`[HBA0]`	Section for each HBA
`WWID=11223344`	Read-only variable
`HostAdapterBiosEnable=1`	Input or current value when read from HBA
`SelectBootEnable=1`	Input or current value when read from HBA
`BootDeviceWWID=22334455`	Input or current value when read from HBA
`BootDeviceLUN[0]=1111`	Input (default LUN) or current value when read from HBA

The text file can be edited to insert the WWID of the boot volume and the LUN number that the HBA will boot from. The utility discovers the WWID of the HBA and writes the information to the text file. The text file presents the options in human readable format. The utility updates the hardware with the boot volume WWID and boot LUN read from the text file.

Example

```
SSTK-v.2.7# ./hplpcfg /se h.ini
SSTK-v.2.7# cat h.ini
[HBA0]
WWID=10000000c94ab9de
HostAdapterBiosEnable=1
SelectBootEnable=1
BootDeviceWWID=50060e801439c901
BootDeviceLUN[0]=1
```

hpqlarep

Synopsis

`hpqlarep`

Description

The `hpqlarep` utility enables or disable the EFI or BIOS boot setting on QLogic 2 GB or 4 GB HBAs.

Options

`hpqlarep /s filename` saves the HBA configuration to *`filename`*.

`hpqlarep /l filename` loads the HBA configuration to *`filename`*

Format

The configuration file for EFI boot setting has the following format:

- For 2 GB HBAs

```
[HBA0]                ; section for each HBA - HBA No
WWID=11223344         ; read-only variable   - HBA WWPN
SelectiveLogin=1      ; input from customer  - 1 (Enable) or 0 (Disable)
EnableSelectiveLUN=1  ; input from customer  - 1 (Enable) or 0 (Disable)
BootDeviceWWNN=2000000000000000 ; input from customer  - Boot Device Node name
BootDeviceWWPN=2200000000000000 ; input from customer  - Boot Device Port name
BootDeviceLUN[0]=0     ; input from customer  - Boot Device LUN ID
```

- For 4 GB HBAs

```
[HBA0]               ; section for each HBA - HBA No
WWID=11223344        ; read-only variable   - HBA WWPN
SelectiveLogin=1     ; input from customer  - 1 (Enable) or 0 (Disable)
EnableSelectiveLUN=1 ; input from customer  - 1 (Enable) or 0 (Disable)
BootDeviceWWPN=2200000000000000 ; input from customer  - Boot Device Port name
BootDeviceLUN[0]=0   ; input from customer  - Boot Device LUN ID
```

NOTE: Boot Target Device Node Name is not needed on 4 GB HBA. If SelectiveLogin and EnableSelectiveLUN are both set to 0, the utility automatically clears out the BootDeviceWWNN/BootDeviceWWPN and BootDeviceLUN (EFI database entries).

hpsetacpi

Synopsis

```
hpsetacpi
```

Description

The `hpsetacpi` tool sets the ACPI mode to Windows or Linux, according to command line parameter informed.

Options

Table 5-15 hpsetacpi Options

Option	Description
Mode	The new ACPI configuration mode. Available choices include Windows, Longhorn or Linux.
`-d`	Displays debug info
`-v`	Displays the version
`-h`	This help menu

The following table details the relation between the `MODE` parameter and `ACPIConfiguration` Mode value.

Mode Parameter	ACPIConfiguration Mode Value
Windows	Windows
Longhorn	Longhorn
Linux	Default or single-pci-domain (the Mode value depends on the machine model.

Example

```
SSTK-v.2.7-8# ./hpsetacpi
ACPI Mode:      single-pci-domain

SSTK-v.2.7-8# ./hpsetacpi linux
ACPI mode was properly set
```

hpshowdisk

Synopsis

```
hpshowdisk
```

Description

The `hpshowdisk` utility helps to identify disks recognized by the system. This utility can be useful when the system has a large number of disks, typically on SAN architectures. Besides device name and size, the utility includes information about disk controller, SAN storage devices WWIDs, and SAN logical disks LUNs.

The `hpshowdisk` utility is internally used by SSTK scripts to display the Available Disks menu during installations. It is also used externally with the Rapid Deployment Pack where, by default, it is called the `showdisk` application. SSTK scripts uses the `-n` option to avoid this external application call.

Options

Table 5-16 hpshowdisk Options

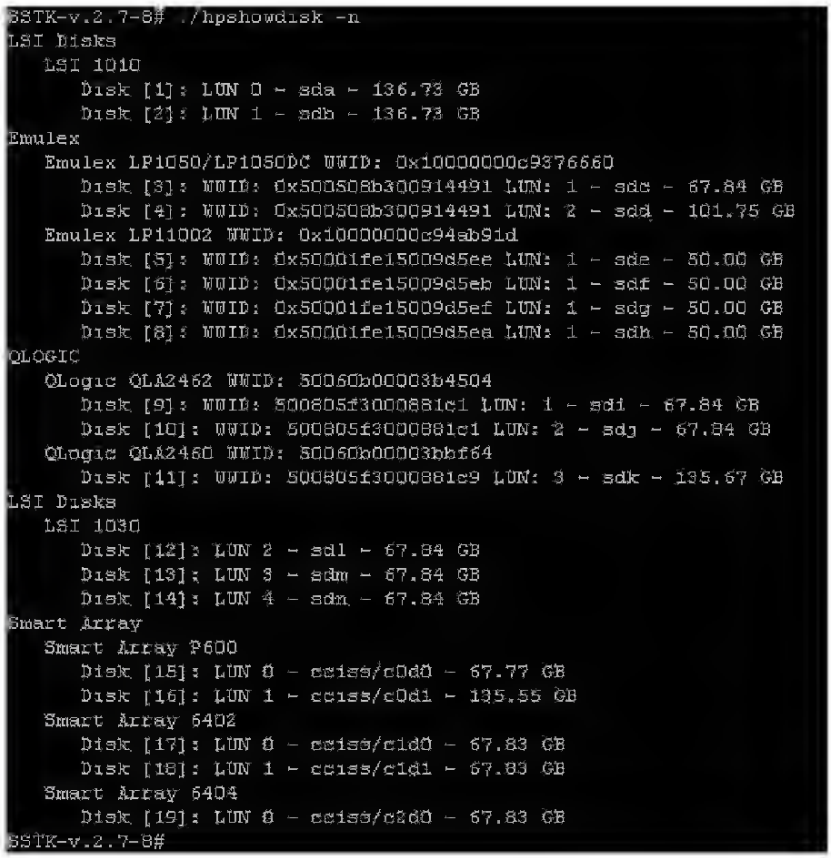
Option	Description
`-v | --version`	Prints the version
`-h | --help`	Prints this help.
`-n | --no-showdisk`	Do not use Altiris RDP `showdisk` as device order.
`-i idx`	Device index.
`-p`	Dumps PCI information for device in format *`vendor`*`:`*`device`*`:`*`svendor`*`:`*`sdevice`*.
`-d`	Dumps device name for device.
`-s`	Dumps the disk size.

Example

The following figure shows an example of `hpshowdisk` output in a system with 19 disks:

- 5 smart array disks (disks 15 to 19)
- 3 SAN disks through QLogic HBA cards (disks 9 to 11)
- 6 SAN disks through Emulex HBA cards (disks 3 to 8)
- 5 LSI disks through Emulex HBA cards (disks 1, 2, 12, 13 and 14)

Figure 5-1 hpshowdisk Utility


SSTK-v.2.7-8# ./hpshowdisk -n
LSI Disks
LSI 1010
Disk [1]: LUN 0 - sda - 136.73 GB
Disk [2]: LUN 1 - sdb - 136.73 GB
Emulex
Emulex LP1050/LP1050DC WWID: 0x10000000c9376660
Disk [3]: WWID: 0x500508b300914491 LUN: 1 - sdc - 67.84 GB
Disk [4]: WWID: 0x500508b300914491 LUN: 2 - sdd - 101.75 GB
Emulex LP11002 WWID: 0x10000000c94ab91d
Disk [5]: WWID: 0x50001fe15009d5ee LUN: 1 - sde - 50.00 GB
Disk [6]: WWID: 0x50001fe15009d5eb LUN: 1 - sdf - 50.00 GB
Disk [7]: WWID: 0x50001fe15009d5ef LUN: 1 - sdg - 50.00 GB
Disk [8]: WWID: 0x50001fe15009d5ea LUN: 1 - sdh - 50.00 GB
QLOGIC
QLogic QLA2462 WWID: 50060b000063b4504
Disk [9]: WWID: 500805f3000881c1 LUN: 1 - sdi - 67.84 GB
Disk [10]: WWID: 500805f3000881c1 LUN: 2 - sdj - 67.84 GB
QLogic QLA2460 WWID: 50060b000063bbf64
Disk [11]: WWID: 500805f3000881c9 LUN: 3 - sdk - 135.67 GB
LSI Disks
LSI 1030
Disk [12]: LUN 2 - sdl - 67.84 GB
Disk [13]: LUN 3 - sdm - 67.84 GB
Disk [14]: LUN 4 - sdn - 67.84 GB
Smart Array
Smart Array P600
Disk [15]: LUN 0 - cciss/c0d0 - 67.77 GB
Disk [16]: LUN 1 - cciss/c0d1 - 135.55 GB
Smart Array 6402
Disk [17]: LUN 0 - cciss/c1d0 - 67.83 GB
Disk [18]: LUN 1 - cciss/c1d1 - 67.83 GB
Smart Array 6404
Disk [19]: LUN 0 - cciss/c2d0 - 67.83 GB
SSTK-v.2.7-8#

hwdisc

Synopsis

```
hwdisc -f[path]filename
```

Description

The hwdisc utility provides an inventory of the server being configured or captured. The hwdisc utility is executed by the server configuration script and captures the following information:

- System Name
- Serial Number
- Asset Tag
- EFI Firmware version
- BMC Firmware version
- MP Firmware version
- Number of Cells
- Total RAM size
- Number of Processors
- PCI devices present in the system

Options

Table 5-17 hwdisc Options

Command line input	Result
-f	This argument specifies the location and name of the hwdisc data file. If [path] is unspecified, the utility assumes that filename is in the current working directory. If no file is specified, the utility generates a file in the current directory utilizing the default name "hwdisc.dat."
hwdisc -f/mnt/data/hwdisc.xml	This command generates the file hwdisc.xml in the /mnt/data location.
hwdisc	This command generates the file hwdisc.dat in the current directory.

Exomple

```
hwdisc -f/mnt/data/hwdisc.xml
```

This command generates the file hwdisc.xml in the /mnt/data location.

A typical file generated by the hwdisc command is similar to the following:

```
<HWDiscovery version="1.0">
<ServerInformation>
<SystemName>Server rx2620</SystemName>
<SerialNumber>52JAPE4343073100</SerialNumber>
<AssetTag>rx2620<AssetTag>
<EFIFw>1.10</EFIFw>
<BMCFw>01.56</BMCFw>
<MPFw>E.02.26</MPFw>
<Cells>1</Cells>
<Ram>512</Ram>
<Processors>2</Processors>
<PCIDevices>
<PCIDevice num="0">
      <Bus>0</Bus>
      <Device>3</Device>
      <Function>0</Function>
      <Id>10024752</Id>
```

```
      <SubID>0E11001E</SubID>
   </PCIDevice>
   <PCIDevice num="1">
      <Bus>0</Bus>
      <Device>4</Device>
      <Function>0</Function>
      <Id>0E11B203</Id>
      <SubID>0E11B206</SubID>
   </PCIDevice>
.
.
.
</PCIDevices>
</ServerInformation>
</HWDiscovery>
```

hwquery

Synopsis

```
hwquery [path]hwdiscfilename [path]allboards.xml variable=string ...
```

Description

The hwquery utility is executed from a script file, in conjunction with other utilities, to control the install process. The hwquery utility uses data from the hardware discovery file in your own scripts. The hwquery utility cannot alter environment variables directly. To set the variable, the output of hwquery must be used by the hosting script. The most common way to use it is to write the output to an intermediate script file that is subsequently called by the hosting script.

Options

Table 5-18 hwquery Options

Command line argument	Description
[*path*]*hwdiscfilename*	This argument specifies the hardware discovery file used to run the query.
[*path*]allboards.xml	This argument specifies the allboards.xmlPCI device list file, which is used to convert PCI IDs found in hardware discovery into device names, such as "Smart Array 6402 Controller."
variable=*string*	In this argument, variable is the name of an environment variable and *string* is a PCI device name or the name of an element from the hardware discovery file. Arguments must be in quotes if *string* contains spaces. The *string* name is case-sensitive.
...	You can specify multiple variable=*string*arguments

Examples

- To produce an hwdisc.xml file that contains <RAM>768</RAM>, use the following input at the command line:

  ```
  hwquery hwdisc.xml allboards.xml MY_SYS_RAM=RAM
  ```

  The hwquery utility produces MY_SYS_RAM=768.

- To produce an hwdisc.xml file that indicates a Smart Array 6402 Controller is present, use the following input at the command line:

  ```
  hwquery hwdisc.xml allboards.xml "TEST=Smart Array 6402"
  ```

  The hwquery utility produces TEST=Smart Array 6402 Controller.

- The hwquery utility is case-sensitive.

  ```
  hwquery hwdisc.xml allboards.xml "TEST=smart array 6402"
  ```

  This command produces TEST=. This behavior is correct even if counter-intuitive. The string is case-sensitive and the argument uses lowercase instead of the title case in the allboards.xml file.

- To produce an hwdisc.xml file that contains <RAM>768</RAM> and <EFIFw>1.10</EFIFw>, use the following input at the command line:

  ```
  hwquery hwdisc.xml allboards.xml MYRAM=RAM MYEFIFW=EFIFw
  ```

  The hwquery utility producesMYRAM=768 and MYEFIFW=1.10.

- If the MYRAM environment variable does not exist, this command creates the variable. To set the value of the <RAM> tag in hwdisc.xml to the MYRAM environment variable use this command:

  ```
  export $( hwquery hwdisc.xml allboards.xml MYRAM=RAM )
  ```

ifhw

Synopsis

```
ifhw [path]hwdiscfilename [path]allboards.xml expression
```

Description

The `ifhw` utility is executed from a script file, in conjunction with other utilities, to control the install process. The `ifhw` utility makes intelligent queries against the hardware discovery file. Queries take the form of a logical expression, and the result of the expression is returned as the exit status of the tool, which the hosting script can use to conditionally perform actions.

Options

Table 5-19 ifhw Options

Option	Description
[path]hwdiscfilename	This argument specifies the hardware discovery file used to run the query.
[path]allboards.xml	This argument specifies the `allboards.xml` PCI device list file, which is used to convert PCI IDs found in hardware discovery into device names, such as "Smart Array 6402 Controller."
expression	This argument specifies the query expression

Examples

The `ifhw` utility may use XML files as input to query the hardware discovery file. The command line input:

```
ifhw hwdisc.xml allboards.xml "PCI:Smart Array 6402"
```

returns the following possible codes:

- 0 (True) if the Smart Array 6402 is present
- 1 (False) if the device is not present
- 2 (Error) if the expression could not be understood
- -1 or 255 if incorrect number of parameters given

The following expressions used as shown in the Synopsis return the indicated result:

- "PCI:Smart Array 6402"

  Returns True if the Smart Array 6402 Controller is found in the system
- HWQ:RAM gte 512

  Returns True if the amount of RAM in the hardware discovery file is at least 512
- HWQ:EFIFw neq "1.10"

  Returns True if the EFI Firmware version in the hardware discovery file is not 1.10
- HWQ:SystemName eq "Server rx4640"

  True if the system name in the hardware discovery file exactly matches "Server rx4640"
- "PCI:Smart Array 6402" or "PCI:Smart Array 6404"

  True if the system contains a Smart Array 6402 Controller or a Smart Array 6404 Controller

Table 5-20 ifhw Operators

Operator or term	Result
`and`	True if both operands are true.
`or`	True if either operand is true.
`gt`	True if the first operand is greater than the second.
`lt`	True if the first operand is less than the second.
`gte`	True if the first operand is greater than or equal to the second.
`lte`	True if the first operand is less than or equal to the second.
`eq`	True if the two operands are equal.
`neq`	True if the two operands are not equal.
`not`	True if the operand is false.
`PCI:`*`string`*	True if a PCI device whose name includes *`string`* is found in the hardware discovery file. *`string`* is case-sensitive.
`HWQ:`*`string`*	The hardware discovery file is searched for *`string`*, and the corresponding value is the value of this term. *`string`* is case-sensitive.
`string`	A literal string, used for comparison.
`number`	A literal number, used for comparison.

parted

Synopsis

```
parted [-v | -i | -s | -h]...
[DEVICE [COMMAND [PARAMETER]...] ...]
```

Description

The `parted` utility manages disk partitions. It allows you to make a GPT partition table and to create all the partitions needed. It is also able to create hidden partitions and change the partition GUID to HPSP, MSRES, MSDATA, and ESP.

Options

Table 5-21 parted Options

Option	Description
`-v`	Displays the version
`-i`	Prompts for user intervention, when necessary
`-s`	Never prompts for user intervention.
`-h`	Displays this help message.

Examples

The following examples illustrate the use of `parted`:

- To check the file system:

  ```
  check MINOR
  ```

- To copy a file system to another partition:

  ```
  cp [FROM-DEVICE] FROM-MINOR TO-MINOR
  ```

- To print general help, or help on COMMAND:

  ```
  help [COMMAND]
  ```

- To create a new disk label (partition table):

  ```
  mklabel PARTITION_TYPE
  ```

  The supported values for *PARTITION_TYPE* are: `gpt`, `bsd`, `mac`, `dvh`, `msdos`, `pc98`, `sun`, `amiga`, and `loop`.

- To make a file system *FTYPE* on partition MINOR:

  ```
  mkfs MINOR FTYPE
  ```

  FTYPE is file system type: `fat16`, `fat32`, `ext2`, and `linux-swap`.

- To create a partition:

  ```
  mkpart PTYPE START END
  ```

  PTYPE is partition type: `primary`, `logical`, or `extended`. For GPT partition table only "`primary`" is used.

- To create a partition with a file system:

  ```
  mkpartfs PTYPE FTYPE START END
  ```

  PTYPE is partition type: `primary`, `logical`, or `extended`. For GPT partition table only "`primary`" is used.

  FTYPE is file system type: `fat16`, `fat32`, `ext2`, and `linux-swap`.

- To move a partition MINOR:

  `move MINOR START END`

- To name a partition MINOR to *NAME*:

  `name MINOR NAME`

- To display a partition table, or a partition:

  `print [MINOR]`

- To exit:

  `quit`

- To recovery a lost partition near START and END:

  `rescue START END`

- To resize a file system on partition MINOR:

  `resize MINOR START END`

- To delete partition MINOR:

  `rm MINOR`

- To select the device to edit:

  `select DEVICE`

- To change a flag on partition MINOR:

  `set MINOR FLAG STATE`

  The relevant flags are:

`boot`	Set ESP Partition GUID.
`hidden`	Hidden partition
`hp-service`	Set HPSP Partition GUID.
`msftres`	Set MSRES Partition GUID.

  The accepted values for *STATE* are "on" or "off".

- To create a new GPT partition table in `/dev/sda device`:

  `parted -s /dev/sda mklabel gpt`

- To create a partition with 100 MB, starting in LBA 0 and finalizing in LBA 100 in `/dev/cciss/c0d0 device`:

  `parted -s /dev/cciss/c0d0 mkpart 0 100`

- To set the ESP partition GUID in the first partition on the table:

  `parted -s /dev/cciss/c0d0 set 1 boot on`

- To set the name of the second partition of the `/dev/cciss/c0d0` partition table to "HP Service Partition" use:

  `parted -s /dev/sda name 2 "HP Service Partition"`

sstkencrypt

Synopsis

```
sstkencrypt
```

Description

The `sstkencrypt` utility is used to encrypt the repository password. After encrypted, it should be set in `elilo.conf`, resulting in a more secure connection over the network.

Example

```
SSTK-v.2.7-8# ./sstkencrypt
Type the repository password:
password
Set the 'encryptpass' parameter in SSTK elilo.conf to:
24732a2730252d75
```

A Support Matrix

This section defines the systems and hardware supported for deployment using SSTK.

Supported systems

- HP Integrity rx1620
- HP Integrity rx2620
- HP Integrity rx2660
- HP Integrity rx4640
- HP Integrity rx3600
- HP Integrity rx6600
- HP Integrity rx7620
- HP Integrity rx8620
- HP Integrity rx7640
- HP Integrity rx8640
- HP Integrity Superdome sx1000
- HP Integrity Superdome sx2000
- HP Integrity BL860c server blade
- HP Integrity BL870c server blade
- HP Virtual Machine systems

Supported LSI and SmartArray SCSI/RAID controllers

Table A-1 SCSI Controllers

Controller	Part Number
SCOWS 1000 U160	A7059A
SCSI 1010 U160	A7060A
SCSI 1030 U320	A7173A
SAS 1068	

Table A-2 ISS Controllers

Controller	Part Number
Smart Array 5302	A9825A
Smart Array 5304	A9826A
Smart Array 6402	A9890A
Smart Array 6404	A9891A
P600	337972-B21
P400	05132-B21
P800	AD335A
E500	AH226A
P400/SB40C	

Supported Emulex and QLogic Fiber Channel controllers

Table A-3 QLogic controllers

Cantroller	Part Number
QLogic 4GB 1P QLA2460	AB429A
QLogic 4GB 2P QLA2462	AB379A
QLogic 4GB 2P QLA2462	AB379B
QLogic 4GB PCIe 1P	AE311A
QLogic 4GB DC PCIe 2P	AE312A
QLogic 4GB Mezz	403619-B21

Table A-4 Emulex controllers

Cantroller	Part Number
FC LP982	A7298A
FC LP9802	AB232A
FC LP1050	AB467A
FC LP1050DC	AB466A
FC Emulex 4GB	AD167A
Emulex 4GB DC	AD168A
Emulex 4GB PCIe 1P	A8003A
Emulex 4GB DC PCIe 2P	A8002A

Supported Intel, Broadcom, and Neterion NIC I/O cards

Table A-5 Intel cards

Card	Part Number
Intel Dual Cu NIC	A9900A
Intel Dual FC NIC	A9899A
Intel PCIe 2P 1Gbe T	AD337A
Intel PCIe 2P 1Gbe SX	AD338A

Table A-6 Broadcom cards

Card	Part Number
Broadcom Cu NIC	A7061A
Broadcom FC NIC	A7073A

Table A-7 Neterion cards

Card	Part Number
10G NIC	AD144A
10G PCIx NIC	AD398A

Supported storage

- SW43*xx*
- MSA30/SW44*xx*
- MSA 50
- MSA 60
- MSA 70
- MSA 600
- MSA 700
- MSA 1000
- MSA 1500
- EVA
- XP
- SB40c